VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Nástroj pro efektivní řízení programátorského týmu

The tool for effective management of a programming team

2017 **Miroslav Lazar**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Miroslav Lazar**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Nástroj pro efektivní řízení programátorského týmu
Tool for Effective Management of a Programming Team

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je implementovat nástroj pro efektivní řízení vývojářského týmu jako nástavbu nad již připraveným komunikačním rozhraním. Nástroj bude formou uživatelského rozhraní umístěn mezi interní firemní systém VENTUS a Team Foundation Server. Zde bude sledovat, řídit a analyzovat veškerou aktivitu mezi těmito dvěma systémy. Sledovanou jednotkou je zde úkol (workitem), jenž se ze systému VENTUS distribuuje jednotlivým členům programátorského týmu.

1. Navrhnout a realizovat aplikační logiku popsané aplikace.
2. Upravit stávající migrační rozhraní pro rozpad úkolů ze systém VENTUS.
3. Vytvořit uživatelské rozhraní pro správu a analýzu takto rozpadlých úkolů (struktury popsané výše) a jejich vazeb s možností řešení problémů s kapacitním plánem, předcházet nesplnění termínu, monitorovat aktivitu členů týmu s predikcí jejich vytížení do budoucna, evidovat změny nad jednotlivými úkoly s dopadem na úkoly z dalších projektů.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Radek Garzina, Ph.D.**

Konzultant diplomové práce: doc. Mgr. Miloš Kudělka, Ph.D.

Datum zadání: 01.09.2015

Datum odevzdání: 28.04.2017

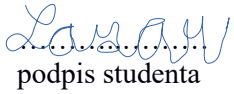
doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *26. 4. 2017*



podpis studenta

Poděkování

Rád bych poděkoval Ing. Radku Garzinovi, Ph.D. za umožnění tvorby této diplomové práce, odbornou pomoc a konzultace v osobním volnu. Doc. Mgr. Miloši Kudělkovi, Ph.D. za konstruktivní kritiku, akademický dohled, trpělivost a lidský přístup. Pracovnímu kolektivu za pochopení a kompromisy nutné pro skloubení práce a studia. Posádce Letiště Frýdlant nad Ostravicí za neustálou motivaci k další práci a svým přátelům horolezcům za osvěžující úniky z reality v pohořích Slovenska.

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.“

Dne: *26. 4. 2017*

.....

podpis zástupce

Abstrakt

Cílem práce je seznámit čtenáře se způsoby řízení vývoje softwarového produktu v rámci střední firmy. Zamyslíme se nad tím, proč je zapotřebí celý výrobní proces naplánovat, nepodcenit a že je to práce pro okruh lidí s mnoha rozdílnými schopnostmi. Jaké jsou nejčastější způsoby práce v rámci programátorského týmu a jaké základní poučky je dobré mít na paměti. Posléze prozkoumáme několik hotových řešení pro řízení týmu lidí podílejících se na společném projektu a pokusíme se zjistit jejich pro a proti. Odhalíme, proč jedno z nejrozšířenějších řešení pro vývoj softwaru jménem Team Foundation Server 2013 v některých situacích svou logikou nevyhovuje organizační struktuře mého zaměstnavatele. Na závěr popíšeme důvody a samotný vývoj nástroje pro odstranění odhalených nedostatků Team Foundation Serveru 2013, rozšířeného o možnost zadávání úkolu z podnikového ERP systému. Většina obsažených informací o chodu firmy a způsobu tvorby softwaru je založena na zkušenostech autora, získaných za dobu tří let ve funkci junior programátora v týmu přibližně 8. lidí.

Klíčová slova

.NET; Ext.NET; Team Foundation Server; software pro řízení týmu

Abstract

The aim of the thesis is to introduce the reader to the ways of managing the development of the software product within the middle company. We will think about why it is necessary to plan, underestimate the entire production process and that it is work for a group of people with many different capabilities, what are the most common ways of working within the programmer team and what basic lesson is good to keep in mind. Finally, we will explore a few solutions to manage the team of people involved in the joint project and try to find out their pros and cons. We will reveal why one of the most popular software development solutions called Team Foundation Server 2013 in some situations does not suit my organizational structure in my logic. Finally, we will describe the reasons and the development of the Team Foundation Server 2013 deployment tool, which is enhanced by the ability to enter a task from a corporate ERP system. Most of the information about running of a company and software developing is based on the author's experience gained over three years as a junior programmer in teams of about 8 people.

Key words

.NET; Ext.NET; Team Foundation Server; team management software

Obsah

Seznam použitých zkratek

Zkratka	Význam
TFS	Team Foundation Server
ERP	Enterprise Resource Planning
SQL	Structured Query Language
C#	C Sharp

Seznam ilustrací a seznam tabulek

Úvod

Výroba softwarového produktu je časově náročný proces. Hlavně z důvodů požadavků na jeho dnešní komplexnost. Dnes již nelze jen tak jednoduše pouze zasednout k pracovnímu stolu a začít tvořit. Celý výrobní proces nového projektu musí být předem důkladně naplánován, prodiskutován rozsah požadovaných funkcionalit a stanoveny reálné termíny, ke kterým se očekává software v daném stavu. Na termíny tlačí jak zaměstnavatel, tak zákazník. Zaměstnavatel by vždy měl mít na pamětí, že dohodnout se se zákazníkem na datu dodání bez rozboru rozsahu prací zavání v lepším případě poškozením pověsti, v tom horším tučným penále. Dnes se na výrobě softwaru podílí celé skupiny lidí, od nichž je vyžadována vzájemná spolupráce. Nelze však očekávat, že všichni členové výroby budou spolu vždy skvěle komunikovat, svou práci odevzdávat včas a nebudou se objevovat nečekaná zdržení. Jako například v případě ztráty dat, výpadku elektřiny, nedostatků použité technologie, problému s hardwarem nebo výpovědi zaměstnance. Navíc často jeden zaměstnanec pracuje na více projektech zároveň. Je-li výrobní plán projektu bez dostatečné rezervy a je po zaměstnanci nečekaně vyžadována práce na projektu jiném, dochází ke zpoždění projektu prvního a celkové riziko nedodržení nasmlouvaných termínů opět roste.

V rámci jedné ostravské softwarové firmy je pro řízení projektu využíváno vlastního ERP softwaru Ventus. Ten je však svou logikou uzpůsoben manažerské činnosti a nevyhovuje potřebám výrobních týmů. Ty naopak využívají řešení Team Foundation Server 2013. Bylo proto navrhnuto vyvinout nástroj pro přenos úkolů a požadavků ze systému Ventus do systému Team Foundation Server. TFS 2013 však také zcela nevyhovuje potřebám jednotlivých výrobních týmů. Bylo proto navrhnuto vytvořit nástroj, který by zjištěné nedostatky TFS 2013 odstranil a přidal další vhodné funkcionality.

Obsahem této práce je tedy v první kapitole shrnout praktiky používané při vývoji softwaru ve firmě středního rozsahu. V druhé kapitole se podívat na pár dostupných softwarových nástrojů pro řízení vývoje softwaru a jejich vlastnosti. Účelem třetí kapitoly je vysvětlit důvody ke vzniku vlastního řešení a navrhnout nové vhodné funkce. Ve čtvrté kapitole provedeme analýzu navrhovaného nástroje a přiblížíme některé detaily implementace.

1. Řízení vývoje softwarového produktu

Před zahájením návrhu jakéhokoliv softwaru je zapotřebí se seznámit s problematikou, kterou bude navrhovaný software řešit. V této kapitole je proto vysvětlen způsob organizace práce výroby softwaru v rámci softwarové firmy.

Realizaci softwaru mají na starosti programátorské týmy výrobního oddělení firmy. Ty se skládají ze skupiny lidí, které spojuje buď práce na daném projektu, technologii (.NET, Android, Sharepoint) nebo jiná odbornost (správa ICT, databáze). Členové, kteří software přímo tvoří, zastávají některou ze tří pozic. Nováčkové a lidé vyžadující častější pomoc s náplní své práce zastávají funkci junior programátora. Ti pracují na implementaci běžných, již analyzovaných požadavcích. Zkušenější kolegové by na něj měli částečně dohlížet, občas zrevidovat jeho práci (kód) a v případě výskytu problému, na který analýza nebrala zřetel, konzultovat další postup prací.

Tito zkušenější pracovníci zastávají pozici Senior programátora a jejich rozsah činností bývá mnohem větší. Kromě samostatného programování bývají účastníky analytických schůzek, kde s tvůrci analýz (konzultanti) konzultují reálnost provedení požadovaných změn z technického hlediska. Někdy analýzy i sami tvoří, protože nikdo nemá o produktu lepší přehled. Rozhodují, s kolika hodinami práce programátorů bude třeba počítat. Pokud se konzultantům nebo projektovým manažerům zdá nákladnost prací příliš vysoká, nabízejí levnější kompromisní řešení.

Nejschopnější ze senior programátorů zastává funkci Team leadera. U Team leadera nezáleží ani tak na technické odbornosti jako spíše na analytickém myšlení a schopnosti komunikovat. To jak v rámci týmu, tak být jeho zástupcem vůči dalším týmům a vedení. Team leadeři se pravidelně schází na produktových poradách. Hlásí svému vedení postup prací na svých projektech. Upozorňují na problémy a rizika vznikající při vývoji. Domlouvají se na formě další spolupráce mezi týmy, pokud jí vyvíjený produkt vyžaduje. Často je totiž vyžadováno propojení několika rozdílných produktů rozličných týmu s vlastní filozofií práce.

Úzce s programátory jsou propojeni konzultanti. Ti také patří do výrobního týmu, ale jsou od programátorů organizačně odděleni. Konzultant až tak nutně nepotřebuje znát technickou stránku produktu, na kterém se podílí (i když je to výhodou), jako spíše jeho logiku a funkčnost. Konzultant komunikuje se zákazníkem, dělá za určitých situací technickou podporu. Pokud si zákazník přeje doplnění nebo změnu funkčnost používaného produktu, konzultant se snaží zákazníkovi navrhnout nejlepší způsob realizace. Požadavek pak převádí do analýz a zadává programátorům.

1.1 Metodika přístupu k vývoji softwaru

Každý programátorský tým se drží nějakého osvědčeného výrobního postupu. Ať už vlastního nebo obecně známého. Těmto souhrnům pravidel a rad se říká metodiky vývoje

softwaru. Metodika má kromě souhrnu postupů, pravidel a nástrojů také definováno jak se má přistupovat k samotné realizaci a údržbě softwaru.

Během historie vzniklo mnoho přístupů, jak vyvíjet a udržovat software. Typickým příkladem staršího přístupu k vývoji je vodopádový model. Ten předepisuje přesnou jednosměrnou posloupnost kroků, které vedou až k finální podobě softwaru. Od zadání požadavků, tvorby návrhu přes implementaci a verifikaci až po údržbu. U vodopádového modelu je kladen důraz na realizaci celého systému najednou. Již ve své době (1970) byl proto označen jako typický příklad chybného, nefungujícího modelu. Kritici tvrdí, že při vývoji softwaru nejsme nikdy schopní před postoupením do dalšího kroku zcela odladit krok předchozí. Mnoho detailů postupně vyjde najevo až v průběhu implementace. Některé věci, které zjistíme, zneplatní náš návrh a my se musíme vrátit zpátky. Celá dosavadní práce tak částečně přijde vniveč. K selhání funkčnosti přístupu také může přispět zadavatel (zákazník), který až po dokončení produktu zjistí, že takto si funkčnost softwaru nepředstavoval. V nejhorším případě zjistí, že jim požadované funkcionality byly implementovány dle zadání, ale nepřináší očekávanou přidanou hodnotu. Na obranu vodopádového modelu je zapotřebí zmínit případy, kdy je jeho filozofie naopak velkým kladem. Pří jeho využití je ale potřeba ke všem krokům přistupovat zodpovědně. Dáme-li si záležet na analýze projektu, o kterém víme, že se nebude v budoucnu již měnit, můžeme už v návrhu odhalit chyby. Ty by nám v pozdějších fázích mohli znemožnit implementaci nebo by úpravy byly mnohonásobně dražší, než když upravíme analýzu. O použitelnosti vodopádového modelu svědčí jeho využívání firmami spolupracující s ministerstvem obrany USA nebo NASA. Tady lze názorně ilustrovat přínos vodopádového modelu. Při vývoji ovládacího softwaru sondy pro vzdálený průzkum je na návrh kladen velký důraz a vše musí být pečlivě promyšleno. Chyba v návrhu potom často znamená neúspěch celé mise. Příkladem může být selhání mise Mars Climate Orbiter, kdy řídicí středisko odesílalo instrukce v anglických mírách, ale sonda očekávala míry metrické. Výsledkem bylo přílišné přiblížení k planetě a shoření v atmosféře.

Je třeba zmínit, že dnes se již v případě zmíněných organizací čistý vodopádový model nepoužívá. Byl nahrazen V-modelem. Ten je (jako většina dalších přístupů) modifikací vodopádového modelu, kdy má každá fáze přiřazenou vlastní testovací fázi.

Nikdy nelze tvrdit, že je nějaký přístup vyloženě špatný. Bývá spíše nevhodně zvolen pro konkrétní projekt. Pro vývoj softwaru bývá vhodnější například inkrementální přístup. Ten kombinuje sekvenční a iterační přístupy. Na začátku jsou definovány nejdůležitější požadavky vodopádovým přístupem, ty jsou zpracovány, předvedeny. Je-li vše v pořádku, navrhne se další funkce a ta je do softwaru zakomponována. Takto se pokračuje až do finální podoby softwaru. Jde tedy o sérii malých vodopádových modelů.

Dnes jsou velkým trendem nejrůznější agilní přístupy. Protože se dokážou nejlépe vyrovnat se specifiky vývoje dnešního softwaru. Ten dnes prochází i po svém nasazení do produkce stálým vývojem, až evolucí. Podoba a jeho funkce se tak po několika letech můžou od původního konceptu velice lišit. Agilní přístup obecně kombinuje ostatní přístupy. Je obecně

vhodný pro projekty s plánovaným stálým rozvojem. Součástí vývoje při agilním přístupu se také stává zadavatel (zákazník). Vývoj je rozdělen do jednotlivých iterací. V každé programátoři pracují jen na dané sadě úkolů/funkcí a analytici zpracovávají zadání pro budoucí iterace a zpracovávají požadavky na změny funkcí vyvinutých v iteracích předcházejících. Vždy na konci iterace jsou nově implementované funkce předvedeny zadavateli. Díky tomu, že vidí ihned jejich funkčnost, může funkci schválit nebo vyžadovat úpravu. Také má právo na rozhodnutí, na kterou funkcionalitu se zaměřit v příští iteraci. Důležitým pravidlem agilního přístupu je nemožnost měnit zadání vyvíjených funkcí během probíhající iterace. Nejznámějšími typem agilního přístupu k vývoji softwaru je Scrum. Z dalších třeba Getting Real.

2. Software pro řízení projektu

Během výrobního procesu je zapotřebí postup prací průběžně sledovat, včas odhalovat možné pozdržení prací, zadávat členům výrobního týmu další úkoly, dávat si pozor na zahlcení zaměstnanců prací. Uřídit výrobní tým jen s papírem, tužkou a osobním kontaktem se stává docela problematické. Jako ve většině oblastí lidské činnosti, tak i při řízení výrobního týmu lze využít mnoha hotových softwarových řešení. V této kapitole si proto projdeme několik z mnoha dostupných softwarových řešení a pokusíme se zjistit jejich klady a zápory. Speciální pozornost je věnována Team Foundation Serveru 2013 pro jeho aktivní využívání mým zaměstnavatelem.

Musíme však mít na paměti, že ani sebedůmyslnější nástroj nám však nepomůže, pokud do něj vkládaná data nebudou odpovídat realitě. Data by proto měla být často aktualizována. Je proto důležité naučit všechny zainteresované osoby disciplíně.

Takových to řešení je samozřejmě spousta. Avšak kolik způsobů řízení týmu, tolik je i nejrůznějšího softwaru. Některé zastupují spíše univerzální řešení, jiné cílí na více na management než na výrobu. Najít nejvhodnější řešení vhodné svou logikou organizace práce pro náš tým nebo celou firmu není jednoduché. Firmy mohou mít nejrůznější organizační struktury a vlastní ověřené pracovní postupy. Firma má potom možnost buď zvolený software používat s jistými výhradami, nebo tzv. „ohnout“ svůj způsob práce, aby odpovídal využívanému řešení.

V tuto chvíli asi není nutné příliš rozvádět úvahu, zda se za příčinou vzniku takovýchto situací neskrývají pouze špatně nastavené výrobní procesy ve firmě. Protože většina softwaru pro řízení výroby je založena na praxí ověřených a důkladně sepsaných postupech.

2.1 BaseCamp

Typickým zástupcem jednoduchého úkolovacího softwaru, jakých je spousta, je BaseCamp [6]. Ten je primárně určen pro několikačlenné týmy, jehož členové se potřebují vzájemně informovat a úkolovat. Koncepce sedí spíše pro kancelářskou činnost, obchod a další spíše netechnické obory. Práci lze rozdělovat dle projektů nebo týmů. Šablona obsahu je vždy stejná.

Tabulka 2.1 Vybrané vlastnosti a jejich zpracování pomocí softwaru BaseCamp

Hlavní objekt	Projekt – Jeden typ. Pokud nově vznikající projekty obsahují stejné úkoly, zprávy, události je možno tvořit vlastní šablony obsahu. Definice termínů v šabloně se uvádí počtem dnů od data založení projektu.
	Tým – Obsahově shodný s projektem, šablony obsahu nelze tvořit.
Komunikace v týmu	Message Board – Informování členů týmu/projektu o důležitých zprávách.
	Automatic Check-ins - Automatické rozesílání dotazů členům týmu/projektu. Lze zvolit jak často a kdy během dne budou dotazováni.
Globální přehled	My Assigments – Globální přehled všech přiřazených úkolů uživateli ze

	sekce To-Dos rozdělené dle týmů a projektů. Veškeré akce provedené členy týmu/projektu lze sledovat v přehledu týmu/projektu. Události se v My Assigments nezobrazují.
Notifikace	Záložka Hey! upozorňuje na všechny s uživatelem související aktivity v týmech/projektech.
	Uživatel je upozorňován na veškeré s ním spojené změny formou e-mailových zpráv.
Soubory	Dock & Files – Úložiště dokumentů společné pro celý projekt/tým s napojením na Google doc.
	Attachment - Soubor jako příloha úkolu.
Úkolování	To-Dos – Úkoly lze rozdělovat do seznamů. Při úkolování členů týmu lze definovat termín splnění nebo vymezit datum zahájení a dokončení. Úkoly však nelze doplnit o další sloupce.
	Schedule – Časová osa, kde jde vidět nadcházející i minulé úkoly členů týmu a založit novou událost (například schůzku). Tu lze importovat do Outlooku.
Klient	Webový, mobilní a desktopový. Desktopový klient pouze načítá webové rozhraní. Nelze pracovat offline.
Licence	Zkušební verze na 30 dní zdarma. Pro výukové účely zdarma. Firmy 99$ měsíčně. Pro neziskové organizace 50% sleva.

2.2 Atlasian Jira

Obsáhlé řešení nejen pro softwarový vývoj [7]. Nabízí několik šablon projektu, tvorbu globálních přehledů, uživatelské sady filtrů s podporou vlastního dotazovacího jazyka, evidenci odpracovaných hodin, instalaci pluginů.

Tabulka 2.2 Vybrané vlastnosti a jejich zpracování pomocí softwaru Atlasian Jira

Hlavní objekt	Projekt – Prostředí dle zvoleno typu šablony, způsobu práce. Development šablony: Scrum, Kanban, Basic. Business šablony: Project, Task, Process,
Komunikace v týmu	HipChat – Samostatný nezávislý nástroj nebo propojený s projektem. Umožňuje tvořit místnosti, provozovat video hovory, integraci s dalšími nástroji.
	Komentáře u úkolů.
Globální přehled	Dashboard – konfigurovatelná plocha pomocí gadgetů. Lze si tak zobrazit svoje úkoly, grafy a další přehledy s využitím předem nakonfigurovaných filtrů.
	Board – scrum nebo kanban tabule. Sloučení přehledu úkolů z různých projektů na jednu obrazovku.
	Issues – Nejčastěji používané úkoly a vyhledávání pomocí JQL dotazů. Dotazy lze ukládat jako samostatné filtry.
Notifikace	Změny na úkolech formou e-mailových zpráv. Možnost konfigurace

	spouštějící události a cílových osob (skupina, role, jednotlivec) Notificaton helper – Ověření nastavených notifikací.
Soubory	Attachment - Soubor jako příloha úkolu.
Úkolování	Issues – Typy podle šablony projektu. Lze přidávat nové typy a sloupce. Zobrazovat historii změn a aktivit. Definovat termín splnění a časovou náročnost. Sledovat volnou kapacitu a vykázaný čas.
Klient	Webový klient.
	Pro vlastní správu serverová verze.
Licence	Zkušební verze na 30 dní zdarma. Možno doplnit moduly Decumentation a Help Desk. Základní verze: Cloud – na měsíc podle počtu uživatelů. Od 10$ pro 10 uživatelů až po 750$ pro 500 uživatelů. Vlastní server – jednorázově. Od 10$ pro 10 uživatelů až po 12000$ pro 250 uživatelů.

2.3 OrangeScrum

Je nástroj zaměřený přímo na správu životního cyklu softwaru [8]. Logika je přizpůsobena pro okruh lidí pracujících na více projektech s důrazem na správu úkolů.

Tabulka 2.3 Vybrané vlastnosti a jejich zpracování pomocí softwaru OrangeScrum

Hlavní objekt	Projekt – Možnost nastavit časovou náročnost projektu a vyhrazený časový úsek. Lze vycházet ze šablony projektu, ta obsahuje úkoly pouze s před nastavitelným názvem a popisem.
Komunikace v týmu	Komentáře k úkolu. Na komentáře lze odpovídat prostřednictvím emailu, odpověď je automaticky přiřazena k úkolu bez nutnosti přihlašování se do systému. Diskuzní místnost je dostupná formou placeného Add-onu.
	Daily Catch-up – Automatické denní dotazování se uživatelů na stav projektu formou e-mailu.
Globální přehled	Dashboard – Zaměřený na projekty. Kolik lidí na nich pracuje, kolik hodin bylo odpracováno a další přehledy. Zobrazuje úkoly s blížícím se termínem splnění a ty u kterých již byl termín překročen.
	Analytics → Resource Utilization – seznam náročnějších úkolů a jejich vykázaný čas.
Notifikace	V grafickém rozhraní při změně související s uživatelem.
	E-mailem při změně související s uživatelem a individuálně na úkolech.
Soubory	Formou přílohy komentáře úkolu. Příloha lze provázat s cloudovými službami. Všechny přílohy lze zobrazit v Files.
Úkolování	Tasks – Rozdělení v projektu na všechny, uživatele, po termínu a další.

	Dělení do skupin. Úkol může být jeden z mnoha typů + definice vlastních. Obsah se nemění. Možno definovat časovou náročnost a termín splnění. Odpracovaný čas lze zpětně vykazovat nebo měřit v reálném čase.
	Analytics – Grafické statistiky úkolů projektu.
	Gantt Chart – Interaktivní časová osa formou ganttova diagramu.
Klient	Webový klient
	Serverová instalace
Licence	Zkušební verze na 30 dní zdarma. Cloud – na měsíc od 9$ pro deset uživatelů až po neomezený počet za 69$ měsíčně. Vlastní server – jednorázově od 2,999$ pro 10 uživatelů až po 7,499 $ pro 50 uživatelů.

2.4 Microsoft Team Foundation Server 2013

Balík nástrojů pro správu životního cyklu softwaru vyvíjeného vícečlenným týmem [2]. Vhledem k jeho provozování mým zaměstnavatelem se na něj podíváme blíže. TFS 2013 nabízí jednak vlastní systém správy verzí, tvorbu sestavení, správu úkolů, tak komunikaci v rámci týmu. Plný potenciál se projeví až po napojení TFS 2013 na Visual Studio. Z prostředí Visual Studia lze číst zadání svých úkolů, obsluhovat správu verzi, žádat ostatní členy projektu o kontrolu kódu nebo spouštět kompilaci zdrojových kódu uložených na serveru. K jeho uložení lze využít přímo úložný prostor TFS 2013 na serveru nebo služeb GITu.

Funkcí, jenž nás zajímá nejvíce, je správa týmu. TFS 2013 se dokáže přizpůsobit několika přístupům k vývoji dle použité šablony při zakládání týmového projektu. Když například zvolíme šablonu pro scrum, lze vytvářet položky do backlogu, tvořit z nich sprinty, zobrazit burn grafy, grafické přehledy. Sprint si lze také zobrazit na typické scrum tabuli a sledovat kapacitu týmu v aktuálním sprintu.

Z požadavků se tvoří úkoly pro členy projektu, v prostředí TFS 2013 známe jako work itemy. Šablonou projektu je také definováno jaké typy work itemů lze tvořit. Typy work itemů se liší sadou polí a obslužným workflow. Logika workflow neboli česky pracovního postupu definuje sadu pravidel pro automatické nastavování polí work itemu. Příkladem může být akce, kdy řešitel přepne stav work itemu na „Dokončeno“. Pokud má work item definováno workflow pro tuto akci, nastaví pole „Datum dokončení“ na aktuální datum a čas.

TFS 2013 počítá s variantou, že ne každému základní konfigurace vyhovuje a umožňuje mnoho možností, jak přizpůsobit jeho funkčnost přání provozovatele. Pro tento případ obsahuje Visual Studio sadu nástrojů pro správu šablon projektů a modifikaci typů work itemů. Pro potřeby komunikace s TFS 2013 pomocí programovacího jazyka obsahuje .NET vlastní knihovny.

Jde skutečně o důmyslný a robustní nástroj. Přesto svou logikou zaměřenou na projekt není vhodný pro práci týmu lidí, který pracuje na několika projektech najednou. Člen takového týmu pokud chce vidět všechny své úkoly, musí si k tomu nadefinovat vlastní dotaz. V TFS 2013 označený jako query. Ten však musí být obsažena v některém z projektů a k jejímu zobrazení je zapotřebí si pamatovat a otevírat projekt, kde je dotaz nadefinována.

Ta samá situace nastává, když si Nadřízený chce zobrazit úkoly určité skupiny pracovníků. Po vytvoření TFS skupiny a nastavení jejich členů musí být v libovolném projektu nastaven požadovaný dotaz. Ten ale s projektem, kde je definován nemusí mít nic společného. Chybí tedy jakýsi globální pohled. Také by se hodilo grafické znázornění úkolů s vyznačenými termíny splnění.

Dalším problémem je izolovanost členů projektu. Člen je brán jako samostatná jednotka. Pokud na tým nahlížím jako Nadřízený, který zadává členům týmu úkoly, není problém některého člena zahltit úkoly v projektu A, i když ta samá osoba má již spoustu práce v projektu B.

Tabulka 2.4 Vybrané vlastnosti a jejich zpracování pomocí softwaru Microsoft Team Foundation Server 2013

Hlavní objekt	Projekt - tři základní šablony s možností přidávání dalších a modifikací stávajících.
Komunikace v týmu	Rooms - Diskuzní místnosti. Každý projekt má vlastní. Také lze tvořit další vlastní místnosti a spravovat členy.
Globální přehled	Queries - Pouze formou nadefinování vlastních dotazů, grafů a jejich následném připnutí na Dashboard projektu.
Notifikace	Alerts - E-mailová upozornění pro projekt dle nastavených filtrů.
	Events - Definice automaticky generovaných zpráv v diskuzní místnosti při změnách v projektu.
Soubory	Attachments - Soubory jako příloha úkolu.
Úkolování	Work - Typy podle šablony projektu. V šabloně projektu lze definovat vlastní typy a sloupce. V závislosti na typu úkolu definovat termín splnění a časovou složitost. V rámci iterace sledovat volnou kapacitu členů projektu.
Klient	Webový klient
	Serverová instalace
Licence	Do pěti uživatelů zdarma. Cloud - na měsíc podle počtu uživatelů. Od 10$ pro 10 uživatelů, 350$ pro 50 uživatelů, 1150$ pro 200 uživatelů.

2.5 Visual Studio Team Services

Během vzniku této práce byl vydán Team Foundation Server 2015. Na něm je zřejmá snaha vypořádat se s některými zde zmíněnými nedostatky verze předchozí. S příchodem

nového ředitele do čela Microsoftu a zavedení otevřenějšímu přístupu k ostatním platformám se dalo očekávat, že se tato změna promítne také do dalších verzí TFS.

Tak se skutečně stalo. TFS 2017 se snaží na vývoj pohlížet opět trochu jinak (zjevná inspirace Atlassian JIRA). TFS přístupný z webového rozhraní byl z Visual Studio Online opět přejmenován, tentokrát na Visual Studio Team Services.

Ve verzi 2017 tak lze konečně na hlavní stránce vidět seznam work itemů přihlášeného uživatele nezávisle na projektu, work itemy, které uživatel sleduje a změny, které na úkolech provedl.

Dotazy a dashboardy je sice nutno stále tvořit uvnitř projektů, ale i zde došlo k pár vylepšením. Projekt může obsahovat více dashboardů s rozdílným obsahem. Na ty je nyní možno přidávat a konfigurovat nejrůznější widgety podle potřeby. Widgety mohou využívat libovolný před připravený dotaz. Ten tak není omezen pouze na připnutí na úvodní stránku projektu. Dotazy dokážou hledat napříč projekty bez nutnosti definování další podmínky. Jeden projekt také nyní může obsahovat více týmů.

Přes všechna tato vylepšení a úpravy se však TFS stále drží své filozofie a tím pádem nevyhovuje potřebám mého zaměstnavatele.

3. Důvody pro tvorbu vlastního řešení

Zmíněný Team Foundation Server je již léta využíván mým zaměstnavatelem. V této kapitole jsou popsány očekávané funkce pro řešení dříve zmíněných nedostatků, doplněné o další navrhovaná zlepšení.

Za léta provozu a vyladění firemních procesů se naplno projevily popsané nedostatky TFS v oblasti správy úkolů. Hlavně z pohledu Nadřízeno, jehož tým pracuje na více projektech. Vzhledem k mottu zaměstnavatele, že se software má vždy přizpůsobit potřebám firmy, bylo navrhnuto vytvořit vlastní nástroj. Ten by s TFS 2013 spolupracoval a doplňoval jej tam, kde nevyhovuje potřebám mého zaměstnavatele. Přidanou hodnotou by byl lehčí přístup k úkolům, které se zaměstnance skutečně týkají. Pohled na úkoly týmu z pozice Nadřízeného a možnost zadávání úkolů uživatelům TFS 2013. Ty by pocházely z podnikového systému VENTUS. U zadaných úkolů by se dohlíželo na včasné plnění termínů.

3.1 Ventus

Ventus [1] je vlastní firemní ERP řešení a jeho funkci lze přímo vysvětlit definicí zkratky ERP. ERP, anglicky Enterprise Resource Planning, česky Plánování podnikových zdrojů nebo někdy též podnikový informační systém, je označení systému, jímž podnik (nebo jiná organizace) za pomoci počítače řídí a integruje všechny nebo většinu oblastí své činnosti, jako jsou plánování, zásoby, nákup, prodej, marketing, finance, personalistika, atd. Každý organizační útvar (oddělení) typicky potřebuje svou vlastní aplikaci schopnou plnit jeho potřeby. S ERP každý útvar takovou vlastní aplikaci dostane, ale je to navíc aplikace, která umí komunikovat a sdílet informace se všemi ostatními v rámci celé organizace. Pojmem ERP se současně označuje i software, který toto vše zajišťuje.

V systému Ventus jsou tedy zabudovány nejrůznější agendy. Od vykazování práce zaměstnanců, účetnictví, až po evidenci zákazníků a jejich objednávek. Důležitou součástí je správa úkolů a požadavků. Jakákoliv práce, kterou je zapotřebí vykonat, by měla být v systému evidována formou úkolu nebo požadavku. Každý zaměstnanec, který se na daném úkolu, požadavku podílí, průběžně vykazuje odpracovaný čas s popisem náplně práce. Konzultant zodpovědný za řízení úkolu, požadavku nebo jeho nadřízení tak mohou evidovat veškerou práci a zaměstnance, kteří se na řešení podíleli. Takto zjistí kolik času resp. Peněz tento úkol, požadavek firmu stál a pokud je celkový čas prací adekvátní k náročnosti řešení, lze tyto prostředky vymáhat po zadavateli ze strany zákazníka. Takový to pohled je výhodný spíše pro vedení, výpočet mezd a další spíše kancelářské oblasti. Už méně pro výrobní oddělení, třídění dle vyvíjených produktů a týmů.

Často vznikají situace, kdy je vyžadován programátorský zásah různých týmů, jenž vytváří svoje softwarové produkty na různých a často jedinečných technologiích v rámci firmy. Obchodní strategie však vyžaduje vzájemnou komunikaci většiny nabízených firemních produktů pro vyšší atraktivitu pro zákazníka. Při implementaci nebo nasazování softwaru do

produkce však vznikají požadavky, jenž jsou závislé na dalších výrobních týmech. Typickým příkladem je požadavek na blíže nespecifikovanou úpravu číselníku v systému A, jenž tyto data nabízí systému B. Pro tým zodpovědný za systém A vzniká úkol na implementaci samotné změny v systému a posléze v rozhraní pro komunikaci. Zároveň však vzniká úkol pro tým systému B na přizpůsobení se provedeným změnám pro zachování kompatibility.

3.2 Navrhované funkce

Takový to požadavek vzniká v systému Ventus. Je zapotřebí požadavek a jeho zadání přenést k lidem z výrobního oddělení, kteří mají změny provést. Nejlépe do systému Team Foundation Server (TFS). Odpadla by tak nutnost programátora hledat zadání požadavku v systému Ventus. Z TFS lze pomocí Visual Studia zobrazit přidělené úkoly se všemi potřebnými detaily. Programátor tak může jednoduše vidět svou pracovní náplň, zadavatele a evidovat odpracovaný čas. Monitorovací a informační část TFS je však svou logikou nevhodná při řízení a monitorování týmů pracujících na více projektech, jak již bylo zmíněno. TFS tak nadále bude zastávat svou úlohu na úrovni projektu.

Mezi oběma systémy bude existovat spojovací článek formou nástroje šitého na míru této situaci, zahrnující různorodé funkce. Primární funkci je umožnit zobrazení úkolů ze systému Ventus a jejich přidělení a odeslání do různých projektů v TFS. Úkoly ze systému Ventus se budou automaticky zapisovat do plánovaného nástroje pomocí již připravené služby. Při zadání úkolu bude kontrolováno, zda zvolený řešitel má v období splnění úkolu volno, aby se úkolu mohl věnovat. Bude se evidovat, kdo úkol zadal, kdy došlo k zahájení řešení atd… Vše se bude evidovat a následně zobrazovat k tomu vytvořeném uživatelském rozhraní. Součástí řešení bude možnost si úkoly dostupné i již zadané filtrovat dle mnoha kritérií nezávisle na projektu. Uživatelé půjdou zařadit do týmů a pro potřeby Nadřízeného následně sledovat jejich práci jako týmu. Součástí týmového přehledu bude časová osa s vyznačenými úkoly a termíny pro ucelenější pohled a plánování. Systém umožní hlídání termínů, upozorňovat na překročení rozpočtu a zpoždění ve výrobě. Samozřejmostí je historie změn.

Zadáním úkolu pomocí našeho nástroje pro správu úkolů vznikne v TFS jeho kopie. Je-li work item (úkol) v TFS posléze modifikován, je zapotřebí změněné sloupce zpětně přenést také do modulu správy úkolů. K tomuto bude sloužit mechanismus, který bude změny na work itemech v pravidelných intervalech kontrolovat.

Běžný zaměstnanec, který často pracuje na více projektech zároveň, musí pro zjištění zadaných úkolů procházet, ať už pomocí webového rozhraní TFS nebo přes Visual Studio, konkrétní projekty. Je-li náplní úkolu práce mimo Visual Studio a přitom chce mít po ruce zadání nebo potřebuje vykázat odpracovaný čas, musí mít jeden z těchto nástrojů stále otevřen. Pro usnadnění práce s přidělenými úkoly je navrhnuto vytvořit malou desktopovou aplikaci, umístěnou v oznamovací oblasti operačního systému. Ta dle doménového jména aktuálně přihlášeného uživatele načte seznam všech jeho nedokončených úkolů. Zaměstnanec si tak bude

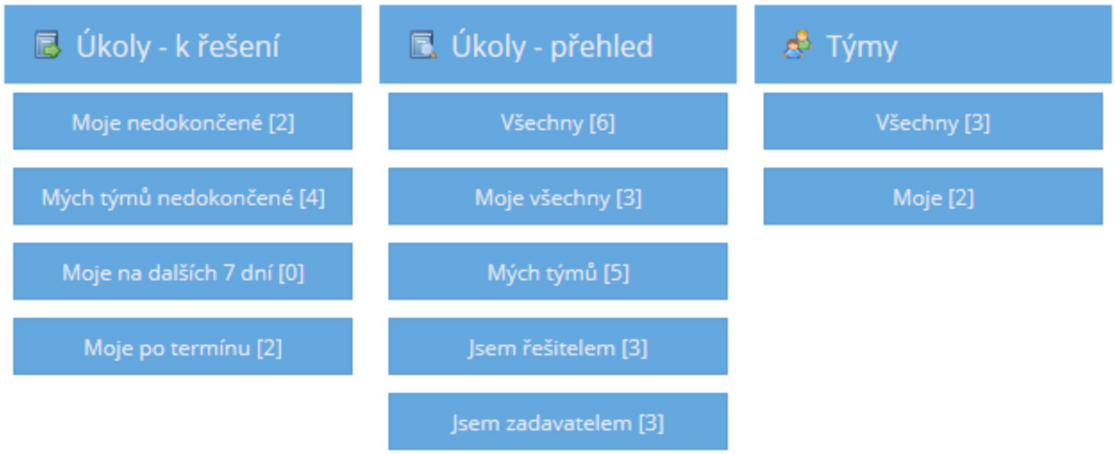
moci zobrazit zadání úkolu, jimž je řešitelem. Měřit čas strávený jeho řešením a ten hned k vybranému úkolu vykazovat/ukládat.

4. Specifikace požadavků

Před zahájením implementace je zapotřebí zpracovat analýzu všech žádaných funkcionalit. V této kapitole jsou důkladně popsány všechny požadované nároky na nový nástroj. Čím důkladněji promyslíme a popíšeme logiku celého řešení, tím snadněji se vyhneme nečekaným zádrhelům při implementaci. Diagramy aktivit vybraných funkcí lze nalézt v příloze 9.A.

4.1 Rozcestník

Proč? Uživatel vyžaduje informace o počtu úkolů rozdělených do přednastavených filtrů podle podmínek v tabulce 1.1 a počtu týmů podle tabulky 1.2. Po zvolení filtru se odpovídající úkoly zobrazí pomocí 3. Přehled úkolů a týmy pomocí 17. Týmový přehled.



Obrázek 4.1: Návrh grafického stylu rozcestníku

Tabulka 4.1 Přehled podmínek přednastavených filtrů pro úkol

Název filtru	**Podmínka pro zobrazení úkolu**	**Zobrazit**
Moje nedokončené	Přihlášený uživatel je řešitelem nebo autorem a stav není nastaven na Dokončeno.	Bez omezení
Mých týmů nedokončené	Řešitel je členem stejného týmu jako právě přihlášený uživatel a stav není nastaven na Dokončeno.	Pouze uživateli v roli Nadřízený
Moje na dalších 7 dní	Přihlášený uživatel je řešitelem, stav není nastaven na Dokončeno a termín splnění vyprší v následujících sedmi dnech.	Bez omezení
Moje po termínu	Přihlášený uživatel je řešitelem, stav není nastaven na Dokončeno a termín splnění je nižší než aktuální datum.	Bez omezení
Všechny	Bez podmínky.	Pouze uživateli v roli Nadřízený

Moje všechny	Přihlášený uživatel je řešitelem nebo autorem.	Bez omezení
Mých týmů	Řešitel je členem stejného týmu jako právě přihlášený uživatel.	Pouze uživateli v roli Nadřízený
Jsem řešitelem	Přihlášený uživatel je řešitelem.	Bez omezení
Jsem zadavatelem	Přihlášený uživatel je zadavatelem.	Pouze uživateli v roli Nadřízený

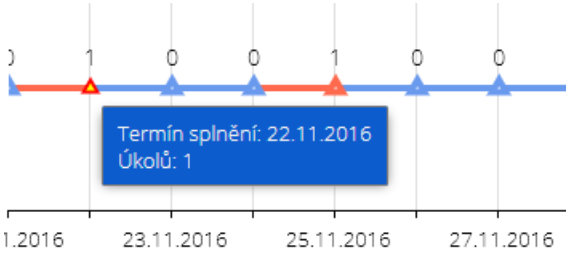
Tabulka 4.2 Přehled podmínek přednastavených filtrů pro tým

Název filtru	Podmínka pro zobrazení týmu	Zobrazit
Všechny	Bez podmínky	Pouze uživateli v roli Nadřízený
Moje	Přihlášený uživatel je členem týmu.	Bez omezení

4.2 Měsíční výhled

Proč? Uživatel vyžaduje být graficky upozorněn na blížící se termíny splnění jemu zadaných úkolů. V případě blížících se termínu půjde tyto úkoly zobrazit.

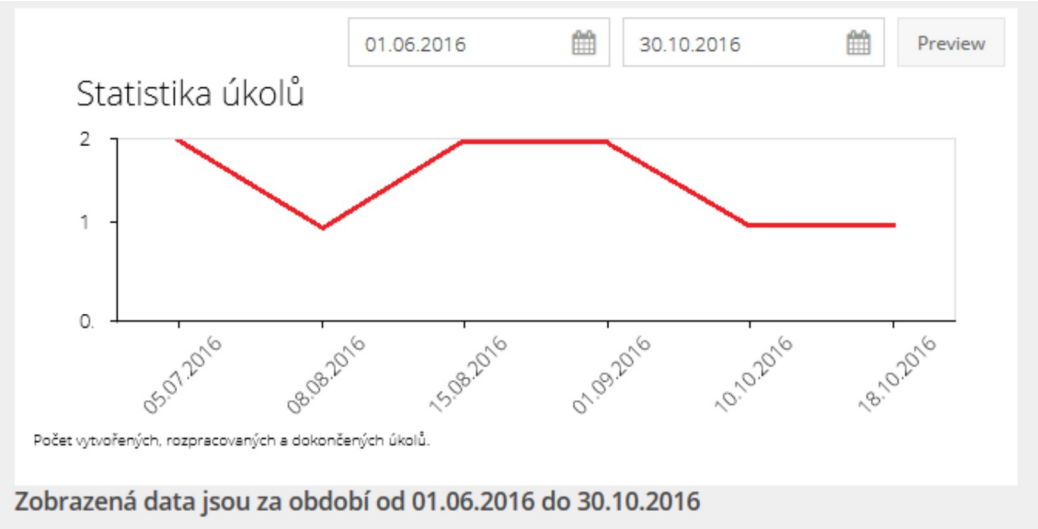
V přehledu pod 1. Rozcestník bude časová osa na měsíc dopředu s vyznačenými termíny splnění úkolů přihlášeného uživatele.



Obrázek 4.2: Navrhovaná podoba měsíčního výhledu

4.2.1 Designové poznámky

- Po najetí myší se zobrazí datum a počet úkolů s termínem splnění v tento den.
- Po vybrání termínu splnění na časové ose se zobrazí seznam úkolů s termínem splnění v tento den díky 3. Přehled úkolů.



Obrázek 4.3: Součástí přehledu bude také graf s časovou osou a znázorněným vývojem vytvořených, dokončených úkolů s možností nastavit od-do

4.3 Přehled úkolů

Proč? Uživatel potřebuje uživatelské rozhraní pro přehled úkolů a filtrovat je dle hodnot sloupců.

4.3.1 Role

Nadřízený – uživatel s vyšší úrovní oprávnění.

Systém – Webová aplikace.

Uživatel – běžný přihlášený uživatel.

4.3.2 Prekondice

Je přijata podmínka požadovaných úkolů.

4.3.2 Základní tok

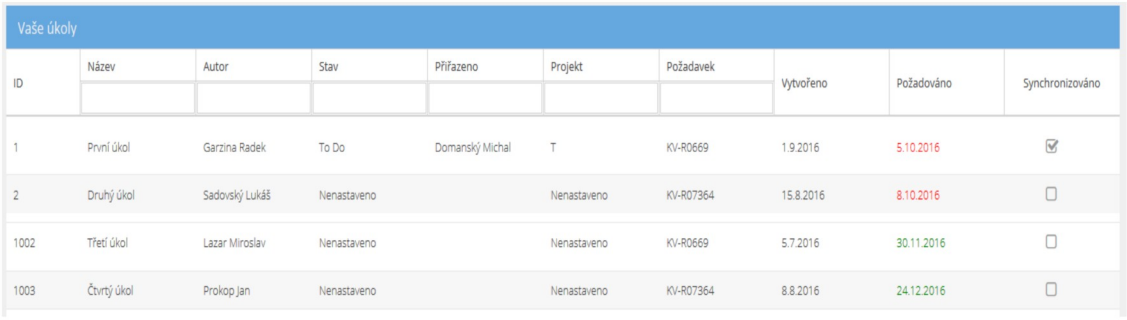
1. Systém načte úkoly.
2. Systém zobrazí sloupce úkolu: Název, Autor, Stav, Řešitel, Projekt, Požadavek, Vytvořeno, Priorita, Požadováno, Synchronizováno.
3. Systém hodnotu pole „Požadováno“ barevně odliší dle jeho data.
4. Systém zobrazí spodní panel seznamu s údajem o celkovém počtu zobrazených úkolů a ovládacími prvky pro nastavení počtu zobrazovaných úkolů na stránku.

4.3.3 Alternativní tok

- Bod 2. Pokud se mají zobrazit pouze nepřiřazené úkoly, budou skryty sloupce: Stav, Řešitel, Projekt.
- Bod 4. Pokud je Uživatel v roli Nadřízený, je panel doplněn o volbu 10. Hromadné zadávání úkolů Řešiteli.

4.3.4 Designové poznámky

- Bod 2. Všechny sloupce budou obsahovat filtr. Formou vyhledávacího pole pro text a kalendářem pro datum. Obsah sloupců také půjde vzestupně nebo sestupně řadit.
- Bod 3. Barvy data pole „Termín splnění“: do týdne – oranžově, po termínu – červeně, jinak – zeleně.


Vaše úkoly

ID	Název	Autor	Stav	Přiřazeno	Projekt	Požadavek	Vytvořeno	Požadováno	Synchronizováno
1	První úkol	Garzina Radek	To Do	Domanský Michal	T	KV-R0669	1.9.2016	5.10.2016	☑
2	Druhý úkol	Sadovský Lukáš	Nenastaveno		Nenastaveno	KV-R07364	15.8.2016	8.10.2016	☐
1002	Třetí úkol	Lazar Miroslav	Nenastaveno		Nenastaveno	KV-R0669	5.7.2016	30.11.2016	☐
1003	Čtvrtý úkol	Prokop Jan	Nenastaveno		Nenastaveno	KV-R07364	8.8.2016	24.12.2016	☐

Obrázek 4.4: Návrh uživatelského rozhraní 4. Přehled úkolů

4.4 Detail úkolu

Proč? Tabulkový přehled neobsahuje všechny informace. Po vybrání úkolu je zapotřebí zobrazit jeho detailní informace.

4.4.1 Role

Uživatel – přihlášený uživatel do Systému.

Systém – Webová aplikace.

Nadřízený – Uživatel Systému s vyšším oprávněním.

4.4.2 Prekondice

Je přijat identifikátor požadovaného úkolu.

4.4.3 Základní tok

1. Systém načte data úkolu a zobrazí pole dle přiložené tabulky.
2. Systém přidá záložku 5. Statistika úkolu.
3. Systém zkontroluje, zda má úkol Řešitele.

4. Systém přidá záložku 6. Informace o projektu.
5. Systém přidá záložku 7. Informace o historii úkolu.

4.4.4 Alternativní tok

- Bod 3. Nemá-li úkol Řešitele → Body 4. a 5. jsou přeskočeny.
- Bod 3. Nemá-li úkol Řešitele a přihlášený uživatel je v roli Nadřízeny → Detail úkolu je doplněn o volbu 8. Zadat úkol Řešiteli.

Tabulka 4.3 Detail položky bude obsahovat tyto sloupce

ID	Id položky
Název	Název úkolu
Popis	Popis úkolu
Stav	Stav úkolu dle TFS
Požadavek	Firemní kód požadavku
Kalkulace	Počet hodin na kolik byl úkol naceněn
Vykázáno	Počet odpracovaných hodin na úkolu
Priorita	Priorita úkolu
Autor	Tvůrce položky
Vytvořeno	Datum vytvoření položky
Termín Splnění	Požadovaný termín splnění úkolu
Původní termín splnění	Původní nemodifikovaný termín splnění
Řešitel	Kdo má úkol řešit
Zadavatel	Kdo přiřadil úkol řešiteli
Zadáno	Datum zadání úkolu řešiteli
Zahájeno	Datum přepnutí do stavu In Progress řešitelem
Splněno	Datum přepnutí do stavu Done řešitelem
Změněno	Datum poslední úpravy
Typ úkolu	Pod jakým typem úkolu je úkol v TFS uložen.

4.5 Statistika úkolu

Proč? Uživatel potřebuje vědět kolik je na úkol vyhrazeno dní od jeho přiřazení. Kolik dní již uplynulo, kolik zbývá, kolik hodin by se mělo denně vykázat, kolik hodin by mělo být vykázáno k dnešnímu dni.

4.5.1 Role

Uživatel – Libovolný přihlášený uživatel systému.

Systém – Webová aplikace.

4.5.2 Prekondice

Je přijat identifikátor úkolu.

4.5.3 Základní tok

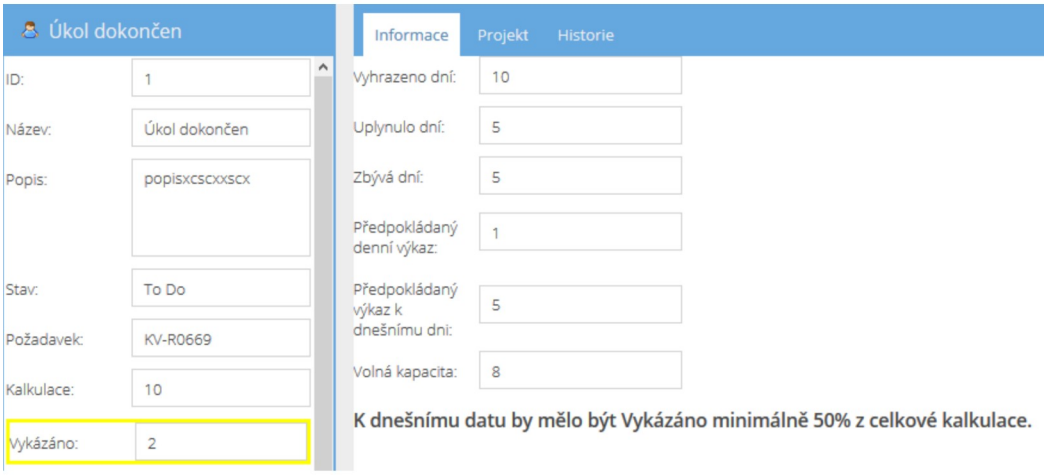
1. Systém načte data úkolu.
2. Systém ze získaných dat úkolu vypočítá požadované informace: Vyhrazeno dní = termín splnění – přiřazeno, Zbývá dní = termín splnění – dnešní datum, Uplynulo dní = dnešní datum – přiřazeno, Doporučený denní výkaz = vyhrazeno dnů / kalkulováno hodin, Doporučený výkaz k dnešnímu dni = uplynulo dnů * doporučený denní výkaz, Volná kapacita = kalkulováno – vykázáno.
3. Systém zobrazí výsledky a skryje nulové nebo nesmyslné hodnoty.

4.5.4 Designové poznámky

- Údaje se budou zobrazovat dynamicky podle údajů úkolu. Například nemá smysl zobrazovat počet zbývajících dnů k řešení u úkolu, který je již po termínu splnění.

Na počítané údaje budou navazovat dodatečné informační zprávy, pokud:

- Není vyplněn údaj Zadáno → Informovat o nutnosti zadání úkolu pro zobrazení dalších údajů. Bez zadaného úkolu nelze další data o úkolu zpracovat.
- Doporučený výkaz k dnešnímu dni < Vykázáno → Zvýraznit žlutě Vykázáno, Informovat v procentech kolik práce by mělo být hotovo.
- Termín splnění < dnešní datum → Zvýraznit červeně Termín splnění, Informovat o uplynulém počtu dní od termínu splnění.
- Termín splnění < Zadáno → Zvýraznit červeně Termín splnění. Informovat o délce mezi Zadáno a Termín splnění.



Obrázek 4.5: Návrh uživatelského rozhraní 5. Statistika úkolu

4.6 Informace o projektu

Proč? Uživatel potřebuje souhrn informaci o projektu, kterého je vybraný úkol součástí.

4.6.1 Role

Uživatel – Libovolný přihlášený uživatel systému.

Systém – Webová aplikace.

4.6.2 Prekondice

Je přijat identifikátor úkolu, úkol má řešitele a je součástí některého z TFS projektů.

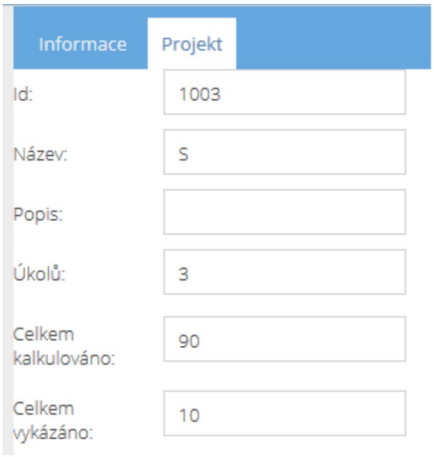
4.6.3 Základní tok

1. Systém načte název a popis projektu, jehož je úkol součástí.
2. Systém zobrazí vypočítané údaje:

   2.1. Počet evidovaných úkolů ze stejného projektu.

   2.2. Celkovou kalkulaci všech úkolů ze stejného projektu.

   2.3. Celkový vykázaný čas všech úkolů ze stejného projektu



Obrázek 4.6: Návrh uživatelského rozhraní 6. Informace o projektu

4.7 Historie úkolu

Proč? Pokud je zadaný úkol v TFS změněn, je zapotřebí o všech takto provedených změnách vést záznamy a zobrazit je v grafickém rozhraní. O tvorbu záznamů historie se stará 12. Aktualizace historie úkolu.

4.7.1 Role

Uživatel – Libovolný přihlášený uživatel systému.

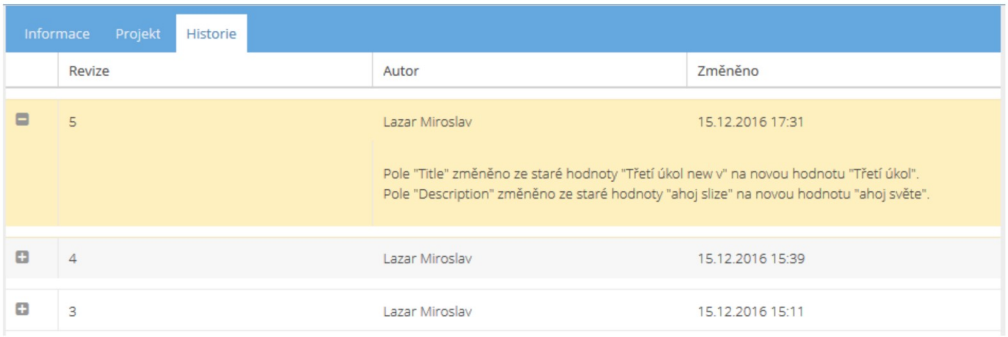
Systém – Webová aplikace.

4.7.2 Prekondice

Je přijat identifikátor úkolu, úkol má Řešitele a je součástí některého z TFS projektů.

4.7.3 Základní tok

1. Systém načte sadu revizí pro vybraný úkol.
2. Systém v tabulkovém náhledu zobrazí ke každé revizi: Číslo revize, Autora změny, Datum a čas změny a rozevírací tlačítko „+“ pro zobrazení podrobností.
3. Uživatel pomocí „+“ zobrazí podrobnost požadované revize, kde budou zobrazeny konkrétní změny. Formát: název změněného pole – stará hodnota -> nová hodnota.



Obrázek 4.7: Návrh uživatelského rozhraní 7. Historie úkolu

4.8 Zadat úkol Řešiteli

Proč? Úkoly ze systému Ventus je zapotřebí zadávat uživatelům TFS. Jako Nadřízený potřebuji nástroj pro úkolování osob v jednotlivých TFS projektech.

4.8.1 Role

Nadřízený – Přihlášený uživatel systémově označen jako osoba s vyšší úrovní práv.

Řešitel – Libovolný uživatel systému pověřený řešením úkolu.

Systém – Webová aplikace.

4.8.2 Prekondice

Je přijat identifikátor zadávaného úkolu.

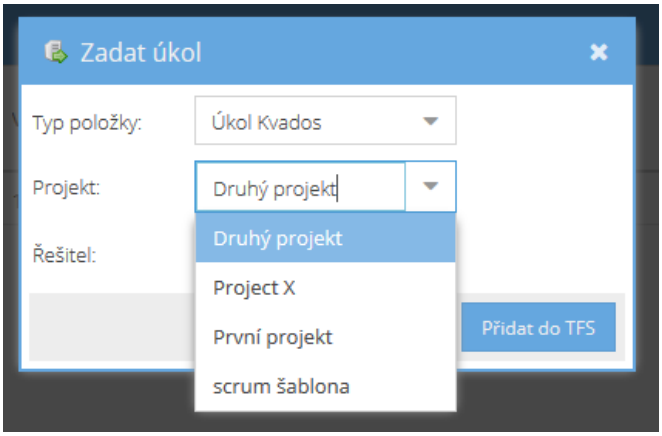
4.8.3 Základní tok

1. Systém zobrazí nabídku typů work itemu, dostupných projektů v TFS a zaměstnanců (řešitelů).
2. Systém zkontroluje, zda termín splnění úkolu < systémové datum.
3. Nadřízený vybere požadované volby.
4. Nadřízený potvrdí volby tlačítkem „Přidat do TFS“.

5. Systém zkontroluje zbývající volnou kapacitu zvoleného Řešitele v daném období pomocí funkce 9. Kontrola volné kapacity Řešitele.
6. Systém vytvoří v TFS nový úkol a nastaví hodnoty polí: Název, popis, stav = To Do, Přiřazeno, Synchronizováno = 1, Kalkulováno, Vykázáno = 0 a další pole podle potřeby.
7. Systém do historie úkolů ve „Správě úkolů“ uloží jeho aktuální stav.
8. Systém do „Správy“ úkolů uloží vazbu na work item z TFS , kdo a kdy zadání provedl.
9. Systém informuje Nadřízeného o úspěšném zadání úkolu a přehled úkolů je aktualizován pro získání aktuálních dat.

4.8.4 Alternativní tok

- Bod 2. je-li úkol po termínu splnění, je Nadřízenému nabídnuto pole pro zadání nového termínu splnění.
- Bod 3. Nadřízený může vybírat pouze z projektů, kterých je v TFS členem.
- Bod 3. Volba Řešitele je následně omezena pouze na členy vybraného projektu.
- Bod 3. Je-li splněna podmínka bodu 2. → Dle vybraného Řešitele a jeho denní kapacity se vyplní pole nový termín splnění s možností editace. Do výpočtu nejsou zahrnuty víkendy. Původní termín se však uloží jako další pole „Původní termín splnění“.
- Bod 4. V případě, že zvolený projekt neobsahuje požadovaný typ work itemu, je na toto Nadřízený upozorněn a proces je přerušen.
- Bod 5. Je-li navrácená hodnota záporná, je Nadřízený informován o možném přetížení Zaměstnance spolu s hodnotou chybějícího počtu hodin pro orientaci. Nadřízený může uložení potvrdit nebo přerušit.



Obrázek 4.8: Návrh uživatelské rozhraní 8. Zadat úkol Řešiteli

4.9 Kontrola volné kapacity Řešitele

Proč? Při přiřazování úkolu je zapotřebí hlídat překročení kapacity daného řešitele, aby na něj nebylo zadáno příliš mnoho úkolů, které vzhledem ke své kapacitě nemůže splnit.

4.9.1 Role

Nadřízený – Přihlášený uživatel systémově označen jako osoba s vyšší úrovní práv.

Řešitel – Libovolný uživatel systému.

Systém – Webová aplikace.

4.9.2 Prekondice

Je přijat identifikátor Řešitele.

4.9.3 Základní tok

1. Systém získá všechny úkoly Řešitele, které nejsou ve stavu „Done“, termínem splnění mezi aktuálním datem a termínem splnění nově přiřazovaného úkolu.
2. Systém od kalkulovaného času těchto úkolů odečte odpracovaný čas. Získá tak čas úkolů, jenž musí řešitel ještě odpracovat (PČ).
3. Systém zjistí počet pracovních dnů mezi dnešním dnem a termínem splnění přiřazovaného úkolu. Počet dní posléze vynásobí denní kapacitou řešitele. Získáme tak čas, jenž by měl řešitel k dispozici, jestliže by pracoval jen na tomto novém úkolů (ČD).
4. Systém vrací ke zpracování výsledek (ČD – PČ).

4.10 Hromadné zadávání úkolů Řešiteli

Proč? Požadavkem na systém je možnost Nadřízené osoby zadávat řešiteli více úkolů najednou v jednom kroku. Bez nutnosti opakovaně spouštět 8. Zadat úkol Řešiteli.

4.10.1 Role

Nadřízený – Přihlášený uživatel systémově označen jako osoba s vyšší úrovní práv.

Řešitel – Libovolný uživatel systému.

Systém – Webová aplikace.

4.10.2 Prekondice

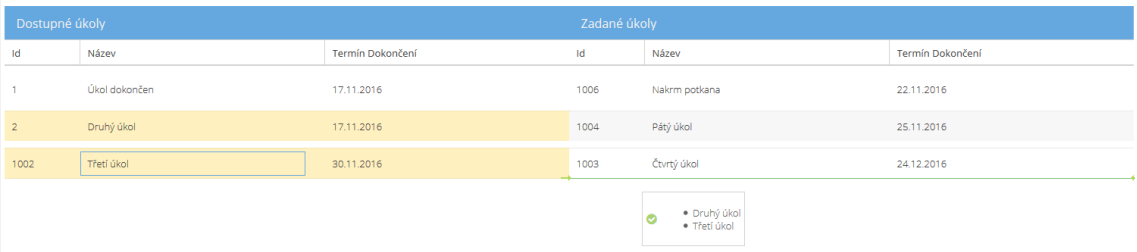
Přihlášený uživatel musí být v roli Nadřízený.

4.10.3 Základní tok

1. Systém zobrazí okna s nepřiřazenými a zadanými úkoly.
2. Nadřízený vybere z levého seznamu úkoly pro přiřazení a přetáhne je do pravého seznamu.
3. Systém pro každý z vybraných úkolů postupuje dle logiky 8. Zadat úkol řešiteli, avšak s tím rozdílem, že zmiňovanou kontrolu dat provede předem pro všechny zadávané úkoly najednou a uložení proběhne až po vyřešení všech konfliktů.

4.10.4 Designové poznámky

- Bod 1. Grafické rozhraní bude řešeno ve stylu souborového manažera s dvěma okny vedle sebe. Levé okno bude obsahovat nezadané úkoly se sloupci Název, Autor, Termín a pravé okno již zadané se sloupci Název, Autor, Stav, Řešitel, Projekt, Termín dokončení. Dvojitým kliknutím na úkol se zobrazí jeho detail.



Obrázek 4.9: Návrh uživatelského rozhraní 10. Hromadné zadávání úkolů Řešiteli

4.11 Aktualizace dat úkolu

Proč? Přenesením úkolu do TFS vznikne jeho oddělená kopie. V případě provedení změn této kopie je nutno tyto změny přenést také zpět do prostředí „správy úkolů“.

4.11.1 Role

Systém – naplánovaná úloha.

4.11.2 Základní tok

1. Systém načte datum a čas poslední provedené synchronizace.
2. Systém se dotáže na work itemy, které byly od posledního data synchronizace změněny v TFS změněny.
3. Systém projde upravené work itemy a porovná je s úkoly ve správě úkolů.
    1. Z TFS se budou v případě změny přenášet pole: Název, Stav, Popis, Vykázáno a další podle potřeby.
    2. Během procesu proběhne pro každý úkol také 12. Aktualizace historie úkolu.
4. Ve správě úkolů se aktualizují úkoly nově zjištěnými hodnotami a atribut poslední změny aktuálním datem a časem.
5. Čas a datum dokončení této aktualizace se uloží jako čas provedení poslední synchronizace.

4.11.3 Alternativní tok

- Bod 2. Není-li nalezen žádný změněný work item, úloha končí.

4.12 Aktualizace historie úkolu

Proč? V rámci prostředí „správy úkolů“ je zapotřebí zobrazovat 7. Historie úkolu změn provedené v TFS.

4.12.1 Role

Systém – naplánovaná úloha.

4.12.2 Prekondice

Je přijat identifikátor úkolu.

4.12.3 Základní tok

1. Systém zjistí číslo aktuální revize úkolu a podle něj revize work itemu s vyšším číslem.
2. Systém vytvoří pro všechny novější revize záznam o revizi a provedených změnách.
3. Systém záznamy o revizi uloží do správy úkolů k jejich pozdějšímu zobrazení.

4.13 Hlídat termín splnění

Proč? Je zapotřebí hlídat blížící se termíny splnění úkolů. Systém by měl zodpovědným osobám zasílat upozornění formou emailů.

4.13.1 Role

Řešitel – Uživatel určený k vykonání zadané činnosti.

Zadavatel – Uživatel, který provedl zadání úkolu řešiteli.

Autor – Uživatel vedený jako tvůrce úkolu.

Systém – Naplánovaná úloha.

4.13.2 Základní tok

1. Systém nalezne nedokončené úkoly, které mají termín splnění za konfigurovatelný počet dní (např. 7).
2. Řešitelům úkolů rozešle email s upozorněním na blížící se termín splnění.

4.13.3 Alternativní tok

- Bod 1. Nejsou-li nalezeny žádné úkoly, úloha končí.
- Bod 2. Nemá-li úkol Řešitele, je upozornění zasláno Autorovi. Nelze-li zjistit e-mailovou adresu Autora, je úkol přeskočen.
- Bod 2. Má-li úkol Řešitele, ale nelze zjistit jeho e-mailovou adresu, je upozornění zasláno Zadavateli.
- Bod 2. Nemá-li Zadavatel e-mailovou adresu, je upozornění zasláno Autorovi. Nelze-li zjistit e-mailovou adresu Autora, je úkol přeskočen.

4.14 Evidence uživatelů

Proč? Pro nutnost skládání uživatelem definovaných týmu, správný chod dalších funkcionalit, nezávislosti na ostatních systémech je potřeba evidovat uživatele Systému.

4.14.1 Role

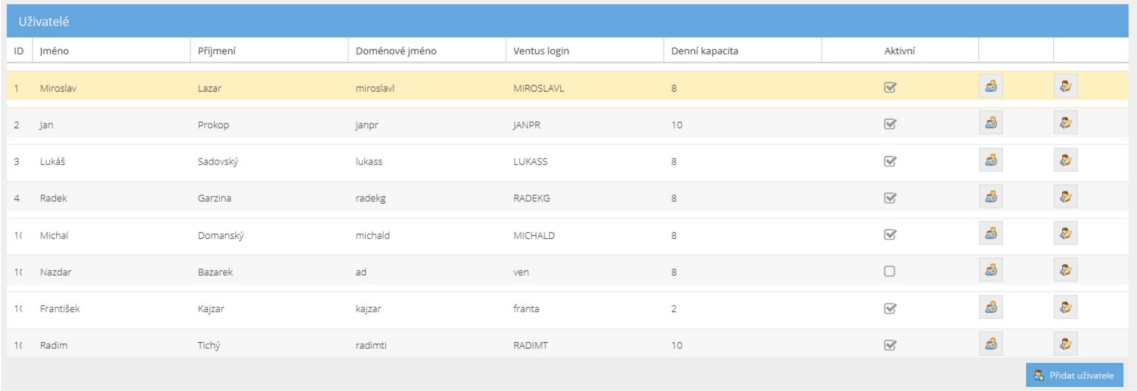
Nadřízený - Uživatel vedený v Systému s vyšší úrovní oprávnění.

Systém - Webová aplikace.

4.14.2 Designové poznámky

- Půjde o evidenci všech zaměstnanců, s kterými se pracuje, nebo se do Systému přihlašují.
- Čtení a zápis bude možný jen pro přihlášeného uživatele v roli Nadřízený.
- Po vybrání uživatele lze záznam upravit nebo zobrazit týmy, kterých je členem viz 15. Evidence Týmů.

- U uživatele se definuje: Jméno, Příjmení, doménové jméno, denní kapacita, e-mail, příznak nadřízený, příznak aktivní.
- Uživatele nelze smazat, pouze označit jako neaktivního.



Obrázek 4.10: Návrh uživatelského rozhraní 14. Evidence uživatelů

4.15 Evidence Týmů

Proč? Uživatele je zapotřebí seskupovat do týmů. Získáme tak týmový pohled bez závislosti na projektu využitý v dalších funkcích řešení. Tyto týmy je tak nutno evidovat a mít možnost definovat jejich složení.

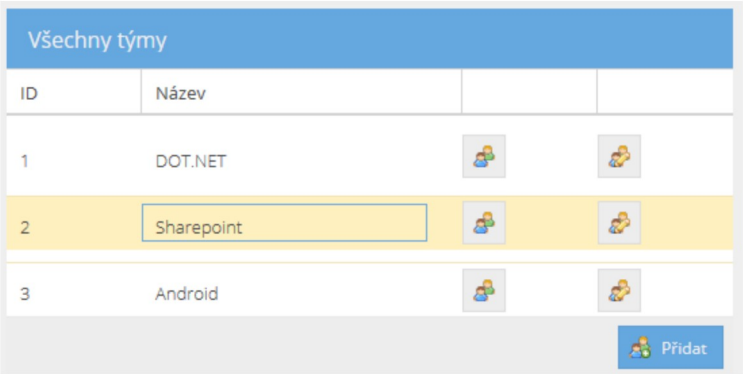
4.15.1 Role

Nadřízený - Uživatel vedený v Systému s vyšší úrovní oprávnění.

Systém - Webová aplikace.

4.15.2 Designové poznámky

- Půjde o seznam týmů, s kterými se v Systému pracuje.
- Čtení a zápis bude možný jen pro přihlášeného uživatele v roli Nadřízený.
- Mazání nebude kvůli vazbám možné.
- Po vybrání týmu bude nabídnuta možnost zobrazit seznam uživatelů, jenž jsou členy tohoto týmu. Při kliknutí na stávajícího člena týmů bude nabídnuta možnost tohoto člena z týmu odstranit. Zároveň lze z rozbalovací nabídky přidat do týmu nového člena.



Obrázek 4.11: Návrh grafického rozhraní 15. Evidence týmů

4.16 Přihlášení do správy úkolů

Proč? K přihlášení do webové části řešení je vyžadováno ověření uživatele pomocí Active Directory.

4.16.1 Role

Uživatel – Osoba zadávající své přihlašovací údaje.

Systém – Webová aplikace, vůči které se uživatel autentizuje.

4.16.2 Základní tok

1. Uživatel vloží své doménové jméno a heslo a odešle žádost o přihlášení.
2. Systém zkontroluje formát přihlašovacích údajů.
3. Systém ověří zadané údaje vůči záznamu v Active Directory.
4. Systém nalezne dle doménového jména tohoto uživatele v seznamu svých uživatelů.
5. Systém nalezeného uživatele načte pro systém jako právě přihlášeného.

4.16.3 Alternativní tok

- Bod 2. Není-li uživatelské jméno ve formátu doménového jména a heslo obsahuje nepovolené znaky, je Uživateli zobrazena výstraha a zpracování se přeruší.
- Bod 3. Odmítne-li Active Directory přijaté údaje. Je uživatel informován o neplatném přihlášení a autentizace přerušena.
- Bod 4. Není-li ověřený Uživatel v Systému nalezen, je informován o nenalezení svého záznamu v rámci aplikace. Je mu doporučeno se obrátit na svého nadřízeného, aby zkontroloval data Uživatele pomocí 9. Evidence uživatelů.

4.17 Týmový přehled

Proč? Primárně uživatel v roli Nadřízený potřebuje mít přehled o složení týmu a práci jeho členů. Jako běžný Člen týmu si můžu zobrazit pouze své naplánované úkoly.

4.17.1 Role

Nadřízený – Uživatel systémově označen jako osoba s vyšší úrovní práv.

Člen týmů – Uživatel, který je členem právě zobrazeného týmu.

Systém – Webová aplikace.

4.17.2 Prekondice

Je přijata kolekce identifikátorů týmů.

4.17.3 Základní tok

1. Systém dle přijatých identifikátorů načte Týmy a zobrazí ovládací prvek pro navigaci mezi nimi.
2. Uživatel vybere požadovaný Tým.
3. Systém ověří, že je přihlášený uživatel v roli Nadřízený.
4. Systém načte všechny členy vybraného týmu a znázorní jejich úkoly na časové ose.
5. Systém pomocí 3. Přehled úkolů načte úkoly, které nemají Řešitele.
6. Systém pod všemi ostatními prvky zobrazí 13. Týmová statistika.

4.17.4 Alternativní tok

- Pokud je přehled zobrazen poprvé, je automaticky vybrán první tým v pořadí a pokračuje se bodem 3.
- Bod 3. Pokud není přihlášený uživatel v roli Nadřízený: Bod 4. - Jsou na časové ose zobrazeny pouze úkoly přihlášeného uživatele. Bod 5. - Tento bod je přeskočen.

4.17.5 Designové poznámky

- Bod 4. Vybráním úkolu na časové ose lze zobrazit 4. Detail úkolu. Kliknutím na Člena týmu je zobrazena 14. Statistika uživatele.

Obrázek 4.12: Návrh uživatelského rozhraní 17. Týmový přehled

4.18 Statistika týmu

Proč? Jeden tým může pracovat na více projektech. Je vhodné sledovat, které projekty to jsou a kolik úkolů je týmu v tomto projektu zadáno. Dále jaký je stav těchto úkolů nezávisle na projektu.

4.18.1 Role

Uživatel – Libovolný přihlášený uživatel Systému.

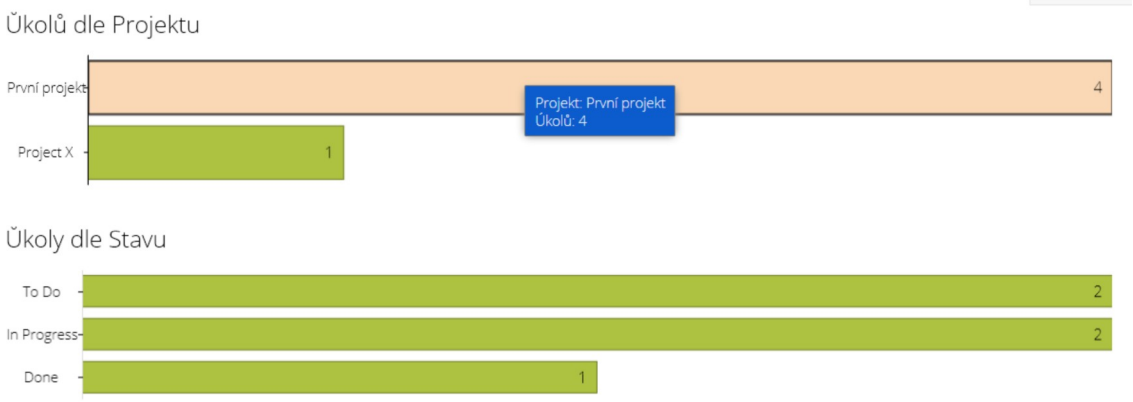
Systém – Webová aplikace.

4.18.2 Prekondice

Je přijat identifikátor týmu.

1. 4.18.3 Základní tok

1. Systém zjistí všechny členy týmu a jejich úkoly.
2. Získané úkoly rozdělí dle jejich Projektů a vykreslí formou sloupcového grafu.
3. Získané úkoly rozdělí dle jejich Stavů a vykreslí formou sloupcového grafu.



Obrázek 4.13: Návrh uživatelského rozhraní 18. Statistika týmu

4.19 Statistika uživatele

Proč? Jeden zaměstnanec může pracovat na více úkolech napříč projekty. Je vhodné sledovat počet jeho úkolů pro jednotlivé projekty a celkové počty zadaných, rozpracovaných a dokončených úkolů.

4.19.1 Role

Uživatel – Libovolný přihlášený uživatel Systému.

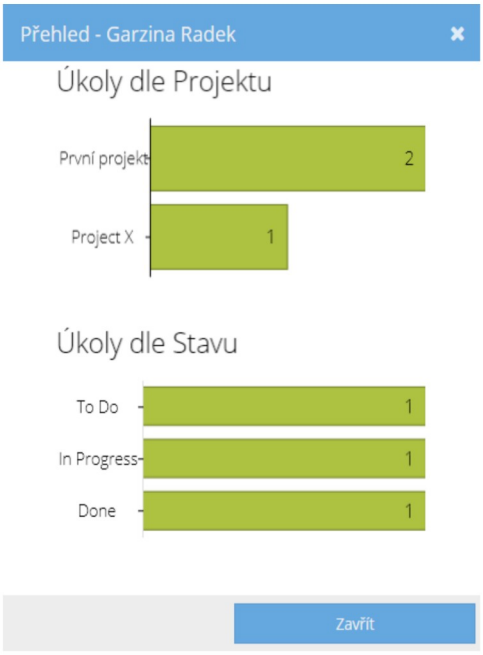
Systém – Webová aplikace.

4.19.2 Prekondice

Je přijat identifikátor uživatele.

4.19.3 Základní tok

1. Dle získaného uživatele Systém načte všechny jeho úkoly, kterých je řešitelem.
2. Získané úkoly rozdělí podle Projektu, pod které spadají a vykreslí formou sloupcového grafu.
3. Získané úkoly rozdělí podle Stavu a vykreslí formou sloupcového grafu.



Obrázek 4.14: Návrh uživatelského rozhraní 19. Statistika uživatele

4.20 Informace o zadaném úkolu

Proč? Zaměstnanec potřebuje být informován o všech svých úkolech bez nutnosti přihlašovat se do jiných dostupných systémů a hledat zadání svých úkolů postupným procházením projektů.

4.20.1 Role

Uživatel – Zaměstnanec.

Aplikace – Navrhovaná desktopová aplikace v oznamovací oblasti OS.

4.20.2 Základní tok

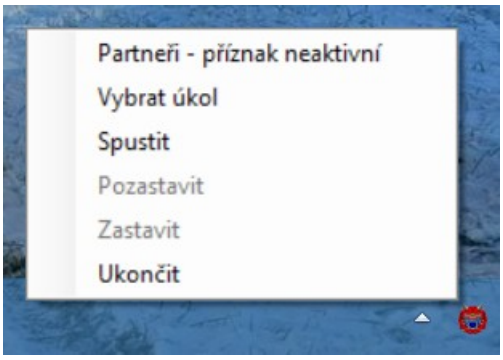
1. Uživatel z kontextového menu vybere volbu „Vybrat úkol“.
2. Aplikace načte a zobrazí ze „Správy úkolů“ seznam všech úkolů, které jsou přiřazeny právě přihlášenému uživatelskému jménu a nejsou ve stavu „Done“ (Dokončeno).
3. Po vybrání úkolu Uživatelem dojde k načtení detailu úkolu a připraví se formulář s detaily úkolu. Zobrazovaná pole budou: „název“, „popis“, „autor“, „zadavatel“, „kalkulace“, „vykázáno“.
4. Systém informuje o zvoleném úkolu formou „bublinové“ informace s názvem úkolu v oznamovací oblasti OS.
5. Systém zpřístupní v kontextovém menu volbu 21. Měřit odpracovaný čas.

4.20.3 Alternativní tok

- Bod 1. Operace čtení úkolů je v případě nedostupnosti vzdálených úložišť ošetřena chybovými hláškami a běh je přerušen.
- Bod 3. Detail úkolu je kdykoliv dostupný přes kontextové menu.

4.20.4 Designové poznámky

- Bod 2. Detail vybraného úkolu bude dostupný z kontextového menu přes volbu s názvem úkolu. Pole jsou pouze pro čtení. Obsah pole „popis“ bude zobrazen po vybrání ve vlastním okně.



Obrázek 4.15: Prototyp uživatelského rozhraní aplikace

4.21 Měřit odpracovaný čas

Proč? Zaměstnanec potřebuje nástroj pro měření, kolik času řešením úkolu spálil. A to aktuálně i celkově. Měření je zapotřebí také pozastavit.

4.21.1 Role

Uživatel – Zaměstnanec.

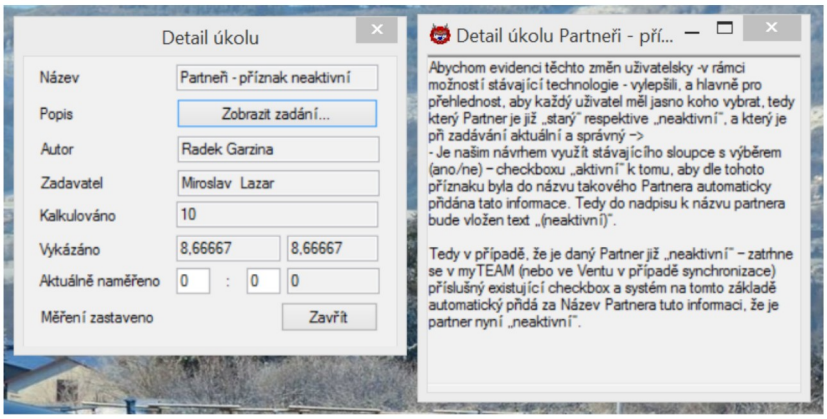
Aplikace – Navrhovaná desktopová aplikace v oznamovací oblasti OS.

4.21.2 Základní tok

1. Uživatel získá 20. Informace o zadaném úkolu.
2. Uživatel spustí měření času.
3. Aplikace informuje Zaměstnance formou „bubliny“ v oznamovací oblasti OS o zahájení měření s informací o doposud vykázaném celkovém času a aktuálně naměřeném času.
4. Uživatel si zvolí možnost „zobrazit detail úkolu“, kde je nové pole s aktuálně naměřeným časem. Formát aktuálně naměřeného času: hodiny, minuty, naměřený čas převedený na celé číslo), vykázaný + nově naměřený čas a informace o probíhajícím měření.
5. Výběrem volby „Zastavit“ Uživatel kdykoliv měření zastaví a Aplikace zobrazí detail úkolu s naměřenými údaji. V detailu se zpřístupní nová volba „Uložit“, která umožní 22. Vykázat odpracovaný čas.

4.21.3 Designové poznámky

- Bod 2. Zpřístupní se volby „Pozastavit“ a „Zastavit“.



Obrázek 4.16: Návrh uživatelského rozhraní detailu úkolu

4.22 Vykázat odpracovaný čas

Proč? Zaměstnanec při vykazování práce u TFS work itemu musí tyto údaje manuálně zadávat přes dostupná grafická rozhraní. Pokud již díky 21. Měřit odpracovaný čas u dříve zvoleného úkolu získal odpracovaný čas, je vhodné tento čas automatický přičíst k již dříve vykázanému času work itemu v TFS. Vhledem k praxi je povolena naměřený čas před uložením editovat.

4.22.1 Role

Uživatel – Zaměstnanec.

Aplikace – Navrhovaná desktopová aplikace v oznamovací oblasti OS.

4.22.2 Základní tok

1. Uživatel provedl 21. Měřit odpracovaný čas.
2. Aplikace umožní v detailu úkolu editovat pole s naměřeným časem. Uživatel naměřený čas dle potřeby edituje.
3. Uživatel v detailu úkolu zvolí volbu „Uložit".
4. Systém aktualizuje celkově vykázaný čas odpovídajícího work itemu v TFS a jeho verzi v systému „správy úkolů". Aktualizovaný záznam ve „správě úkolů" také rozšíří o hodnotu skutečně naměřeného času bez možnosti editace.
5. Systém informuje uživatele o úspěšném uložení/vykázání, naměřený čas je resetován a okno detailu položky zavřeno.

4.22.3 Alternativní tok

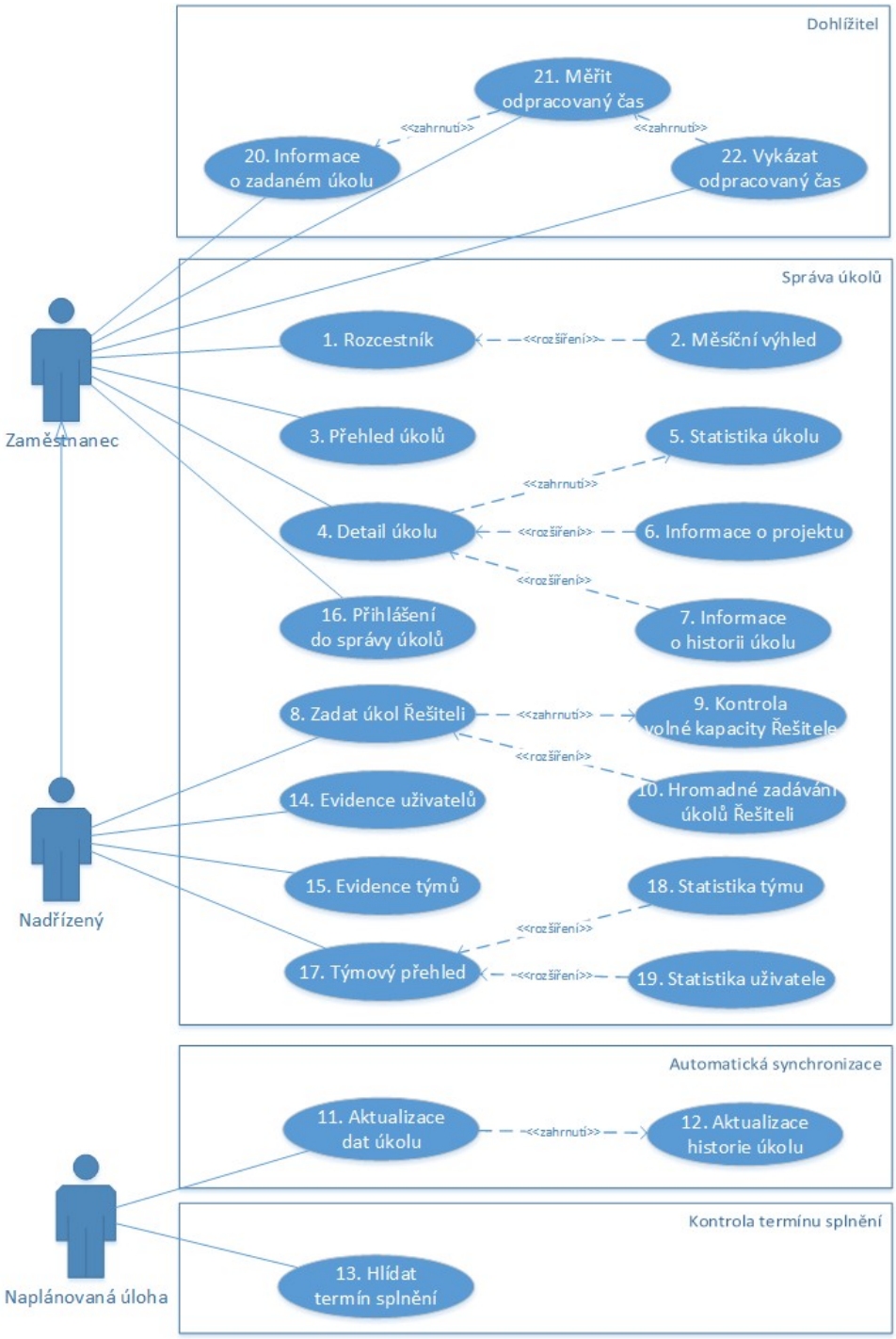
- Bod 3. V případě nedostupnosti vzdálených úložišť se zobrazí chybová hláška a proces se přeruší.

4.22.4 Designové poznámky

- Bod 2. Při úpravě naměřeného času se automaticky aktualizuje jeho celočíselná hodnota a nový celkový výkaz.
- Bod 2. Pokud pole nového celkového výkazu překročí kalkulaci, toto pole zčervená.
- Bod 4. Skutečně naměřený čas je uchováván pro možnou pozdější analýzu.

5. Implementace

5.1 Diagram případů užití



Obrázek 5.1: Diagram případů užití celého navrhovaného řešení

5.1.1 Aktéři

- Zaměstnanec – Libovolný uživatel vedený v modulu správy úkolů.
- Nadřízený – Vedoucí týmu nebo Zaměstnanec pověřený zadáváním úkolů nebo správou nástroje.
- Naplánovaná úloha – Automaticky spouštěna v nakonfigurovaných intervalech.

5.2 Moduly řešení

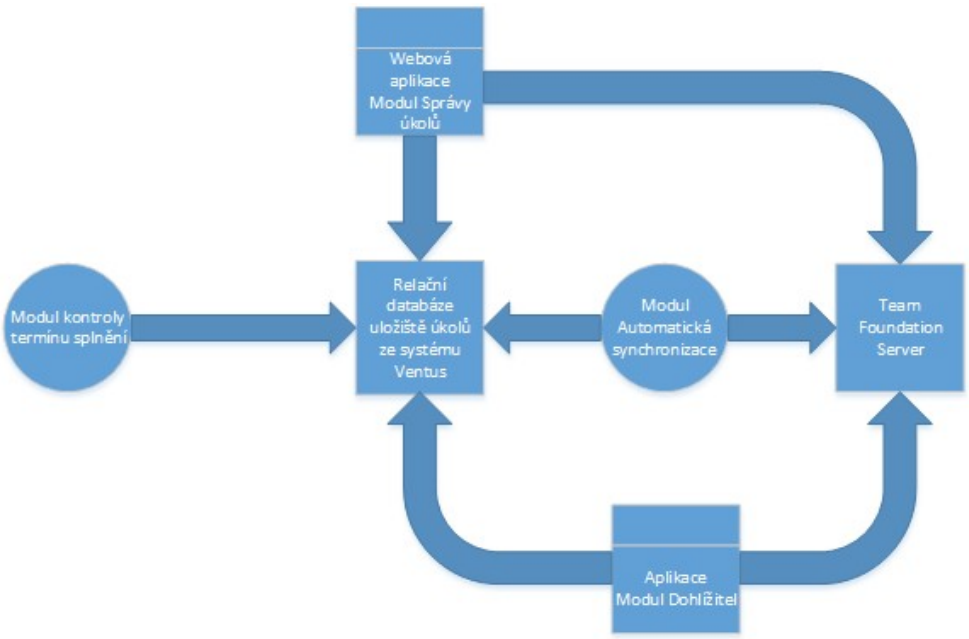
Celé navrhované řešení se dá rozdělit několika samostatně funkčních modulů.

Hlavním modulem bude „správa úkolů“ postavená převážně na úkolech z vlastního úložiště. Půjde o webovou aplikaci umístěnou na serveru dostupnou v rámci firemní sítě. Webová aplikace bude realizovat: 1. Rozcestník, 2. Měsíční výhled, 3. Přehled úkolů, 4. Detail úkolu, 5. Statistika úkolu, 6. Informace o projektu, 7. Historie úkolu, 8. Zadat úkol Řešiteli, 9. Kontrola volné kapacity Řešitele, 10. Hromadné zadávání úkolů Řešiteli, 14. Evidence uživatelů, 15. Evidence týmů, 16. Přihlášení do správy úkolů, 17. Týmový přehled, 18. Statistika týmu, 19. Statistika uživatele.

Pro realizaci funkcí: 11. Aktualizace dat úkolu, 12. Aktualizace historie úkolu bude připraven modul „Automatická synchronizace“. Bude řešený formou Windows služby spuštěné na serveru s přístupem jak do „správy úkolů“, tak TFS. Služba bude spouštěna v před definovatelném intervalu (například 5 minut).

Blížící se termín splnění na úkolech bude hlídat modul „Kontroly termínu splnění“. Půjde o další Windows službu, která se bude automaticky spouštět jednou denně. Modul realizuje funkci: 13. Hlídat termín splnění.

Modulem pro instalaci výhradně na pracovní stanice zaměstnanců bude „Dohlížitel“ neboli „Overseer“. Půjde o desktopovou aplikaci umístěnou na pracovních stanicích uživatele. Ovládána bude z oznamovací oblasti OS. Realizuje funkce: 20. Informace o zadaném úkolu, 21. Měřit odpracovaný čas, 22. Vykázat odpracovaný čas.



Obrázek 5.2: Schéma rozdělení do modulů

5.3 Způsob vývoje

Vývoj probíhal formou týdenní iterace, formou inkrementálního přístupu. Na začátku byl definován BackLog se základními požadavky na celé řešení. Z těch se podle priority vybíraly zadání pro aktuální týden a průběžně přibývaly další.

Před zahájením každého dalšího týdenního cyklu proběhla konzultace se zadavatelem o vyvinutých řešeních za poslední týden. Některé funkce se vracely do produkce pro přiblížení se ideálu i opakovaně. To z toho důvodu, že samotná analýza neodhalí všechna úskalí vývoje a použitelnost řešení se projeví až po implementaci. Často byl napřed vytvořen pouze prototyp, který ukázal, jestli má zvolený způsob řešení smysl. Až po jeho schválení a získaných zkušenostech se začalo plně pracovat na finální podobě.

Vývoj se v první fázi držel vzoru třívrstvého modelu, tedy definováním datové vrstvy a jejího rozhraní. Až po dokončení těchto základů se přešlo k pracím na logické a prezenční vrstvě. Takový to způsob práce byl však možný jen při tvorbě základní kostry. S dalšími iteracemi a doplňovanými funkčnostmi již byly jednotlivé vrstvy modifikovány podle potřeby souběžně. Za celou dobu vývoje se ale na oddělení logické a datové vrstvy pamatovalo.

5.4 Použité technologie

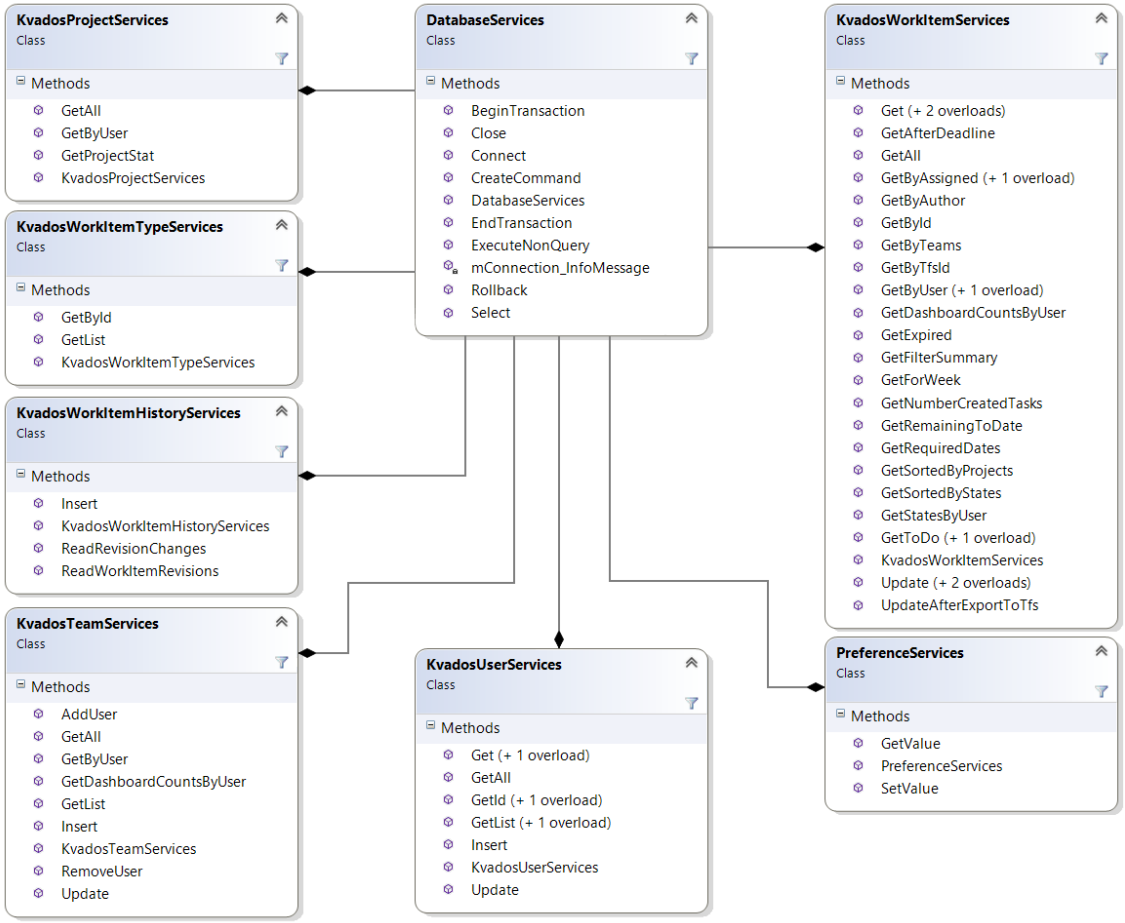
Pro komunikaci s TFS bylo využito jeho API. To je postaveno na technologii .NET stejně jako celé TFS. Vývoj proto také probíhal na této technologii. Permanentním úložištěm se stal SQL Server. Grafické rozhraní využívá webového frameworku ASP.NET doplněného o

.NET framework Ext.NET. Tam, kde Ext.NET není nutně zapotřebí, je využito možností Bootstrapu.

- Ext.NET je framework pro ASP.NET [3] umožňující snadnější tvorbu grafické části informačního systému a programátor se může více soustředit na jeho logiku. Obsahuje velkou paletu tzv. Controls nejčastěji využívaných při zobrazování dat a také přináší do webového prostředí plno funkcí známých spíše z desktopů. Jako je například přenášení dat pouhým „přetáhnutím“ položky do jiného okna. Celé řešení je založeno na rozsáhlé javascriptové knihovně ExtJS, která je mnohem známější než zde využívaný derivát pro .NET. Ext.NET si však pro obsluhu z výchozího ASP.NET bere pouze to nejzákladnější a vyžaduje pro své plné využití pochopení mnoha vlastních atributů a javascriptových funkcí. Ty jsou potřeba primárně pro specifikaci přenášených dat do .NET kódu a pro úpravu vzhledu vygenerované HTML stránky na základě zobrazených dat. Jako podpora při vývoji slouží oficiální stránky ext.net s velkou zásobou řešených příkladů i obsáhlým fórem. Bonusem navíc je potom kniha Ext.NET Web Application Development [5]. Autoři se chlubí delším seznamem firem využívajících tohoto frameworku. Jako například Canon, Disney, Good Year, VMWare… Poslední dobou lze však pozorovat stagnaci dalšího vývoje.
- Bootstrap je sada návrhářských šablon ve formě CSS tříd pro snadnější návrh rozložení stránky, typografie, formulářů, tlačítek a dalších HTML elementů.

5.5 Třídy pro pro přístup k datům

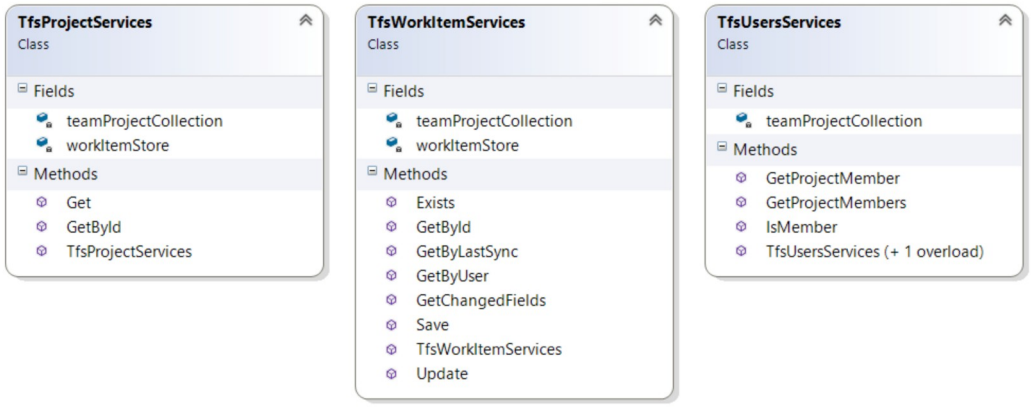
Modul „správy úkolů má vlastní databázové úložiště, s kterým pracují i další moduly. Pro jednotný přístup k tomuto úložišti jsou definovány třídy a metody. Ty zastupují funkci data mapperu. Ten je řešen formou: entita (Tabulka) = objekt třídy. Ke komunikaci s SQL Serverem je využito služeb ADO.NET. Metody dotazy pro požadované funkce samy netvoří. Pouze volají uložené procedury v databázi a čtou jimi navrácená data.



Obrázek 5.3: Diagram tříd pro komunikaci s databází

5.6 Třídy pro komunikaci s Team Foundation Server

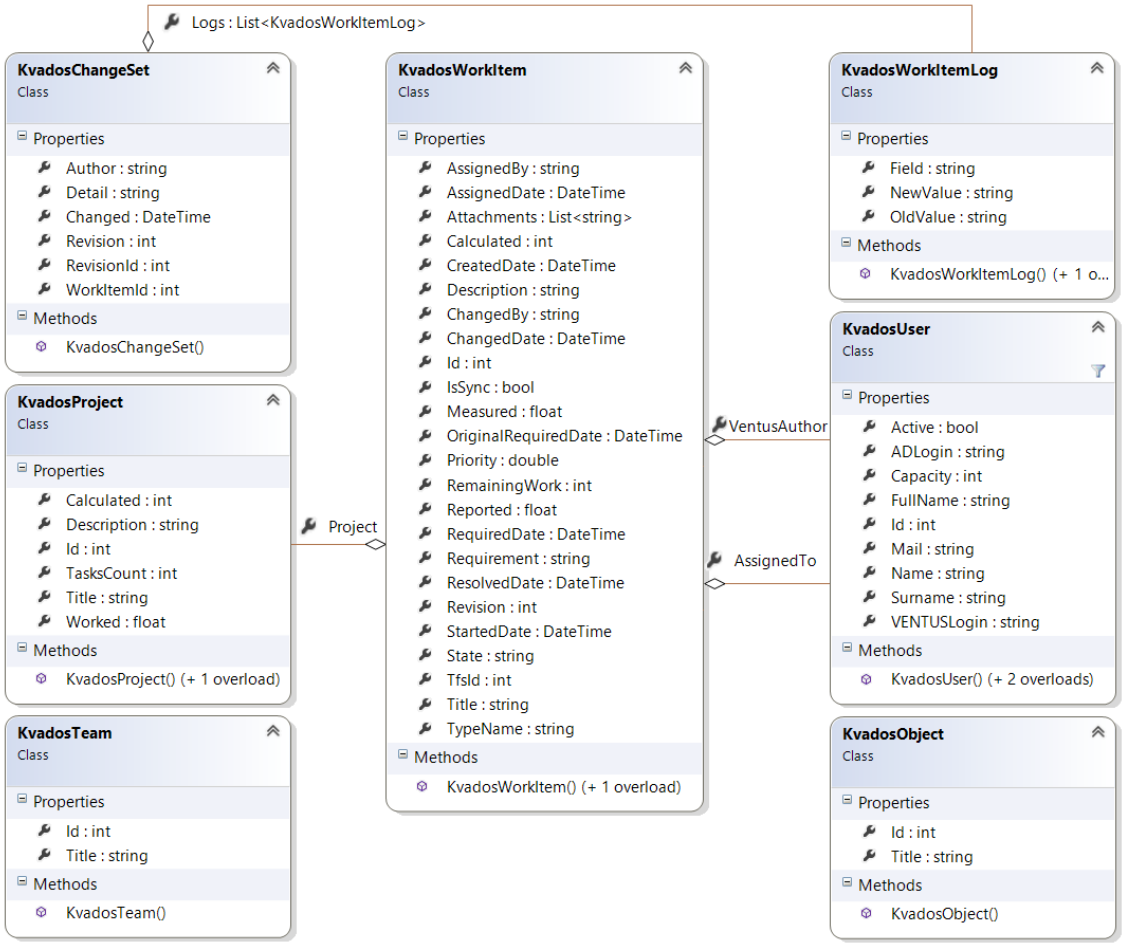
Protože je zapotřebí komunikovat také s TFS, jsou k tomuto účelu definovány vlastní třídy. Ty pro čtení a zápis využívají objektového modelu TFS [4], který je dostupný v rámci knihoven .NET po přidání vhodných referencí.



Obrázek 5.4: Diagram tříd pro komunikaci s Team Foundation Server

5.7 Objektový model

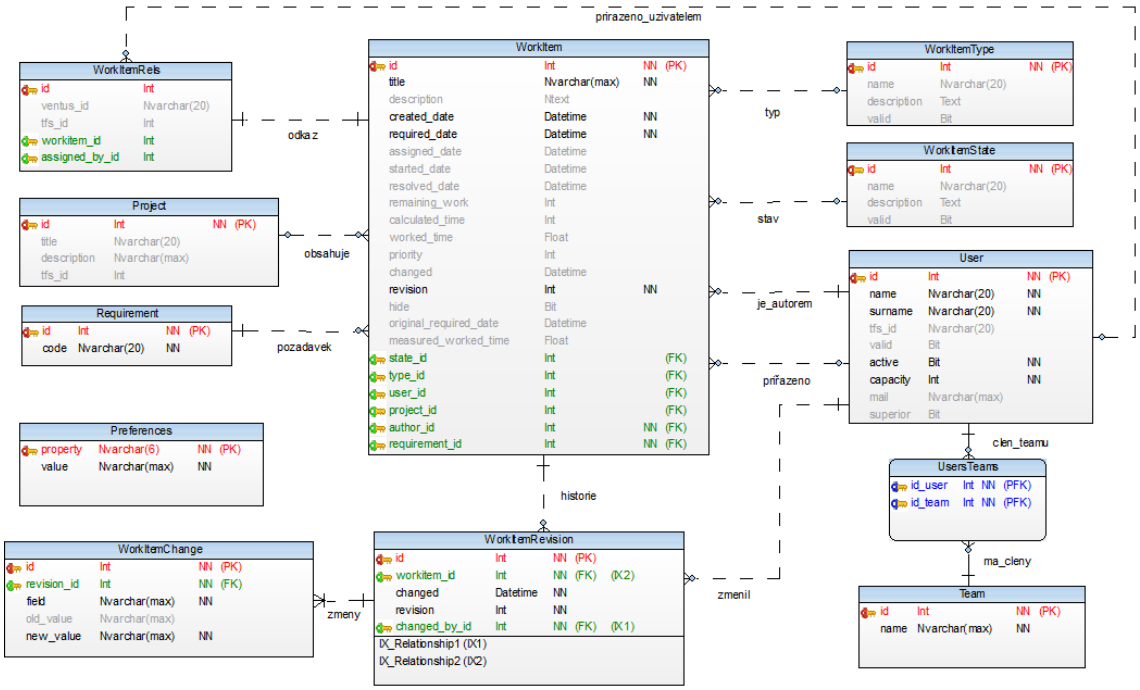
Třídy pro přístup k datům v databázi a TFS naplňují daty třídy objektového modelu. Z takto připravených objektů posléze čerpají údaje prvky grafického rozhraní nebo další funkce bez závislosti na úložišti.



Obrázek 5.5: Diagram tříd objektového modelu

5.8 Datová vrstva

Primárním úložištěm úkolu je relační databáze běžící na SQL Serveru. Do této databáze se automaticky ukládají úkoly ze systému Ventus. Veškeré kontrolní a informační funkce celého řešení čerpají svá data právě odsud.



Obrázek 5.6: Fyzický model systému

Hlavní entitou je zde WorkItem (Úkol). Na něho se vážou další entity Projekt, Požadavek, Typ úkolu, Stav úkolu, Uživatel. Tyto entity mají vlastní tabulku kvůli vazbě 1:N, protože například jeden kód požadavku může být shodný pro více úkolů. Dle zažitých standardů není vhodné textovou hodnotu, která představuje totožnou položku reálného světa opakovat v atributu jednotlivých záznamů.

Historie změn na úkolech je řešena pomocí entit WorkItemRevision což jsou jednotlivé verze (revize) úkoly a ty obsahují záznamy o změnách na jednotlivých polích.

Entita User má více vazeb na úkol z důvodu, že jednou může být uživatel označen jako řešitel úkolu, jindy jako jeho autor. Na uživatele je přes vazební tabulku napojena entita Team. Díky tomu může jeden uživatel být členem více týmů a samozřejmě více uživatelů členem jednoho týmu.

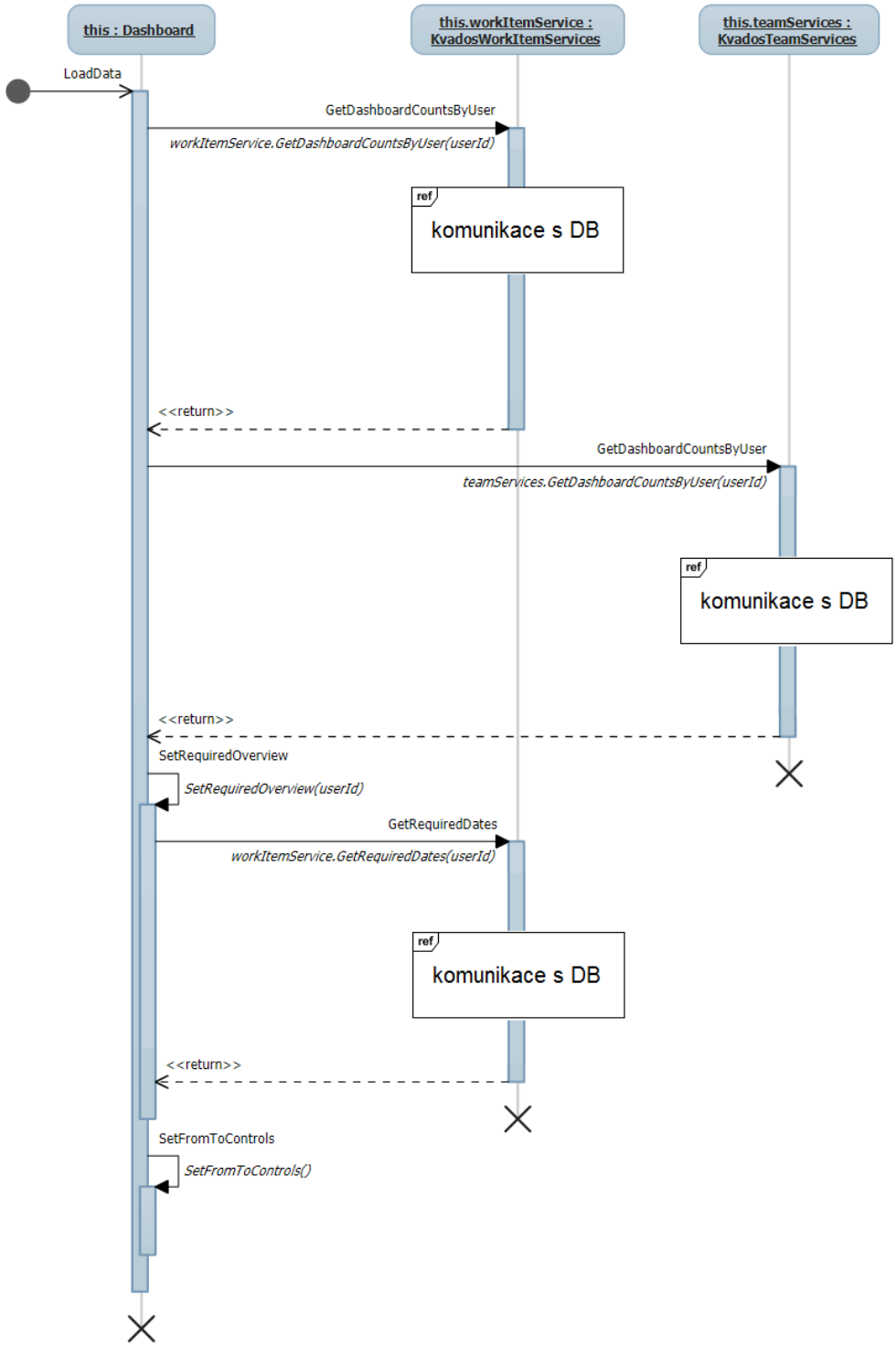
Pro identifikaci úkolu mezi systémy je připravena zvlášť entita WorkItemRels. Ta uchovává id totožného úkolu v ostatních systémech spolu s údajem, kdo tuto vazbu vytvořil.

Čtení a zápis bude řešen formou uložených procedur. Díky uložení přímo v SŘBD získáme jednotné rozhraní pro přístup k datům bez závislosti na dalších aplikacích. Rozhraní tak půjde využít i v budoucích aplikacích.

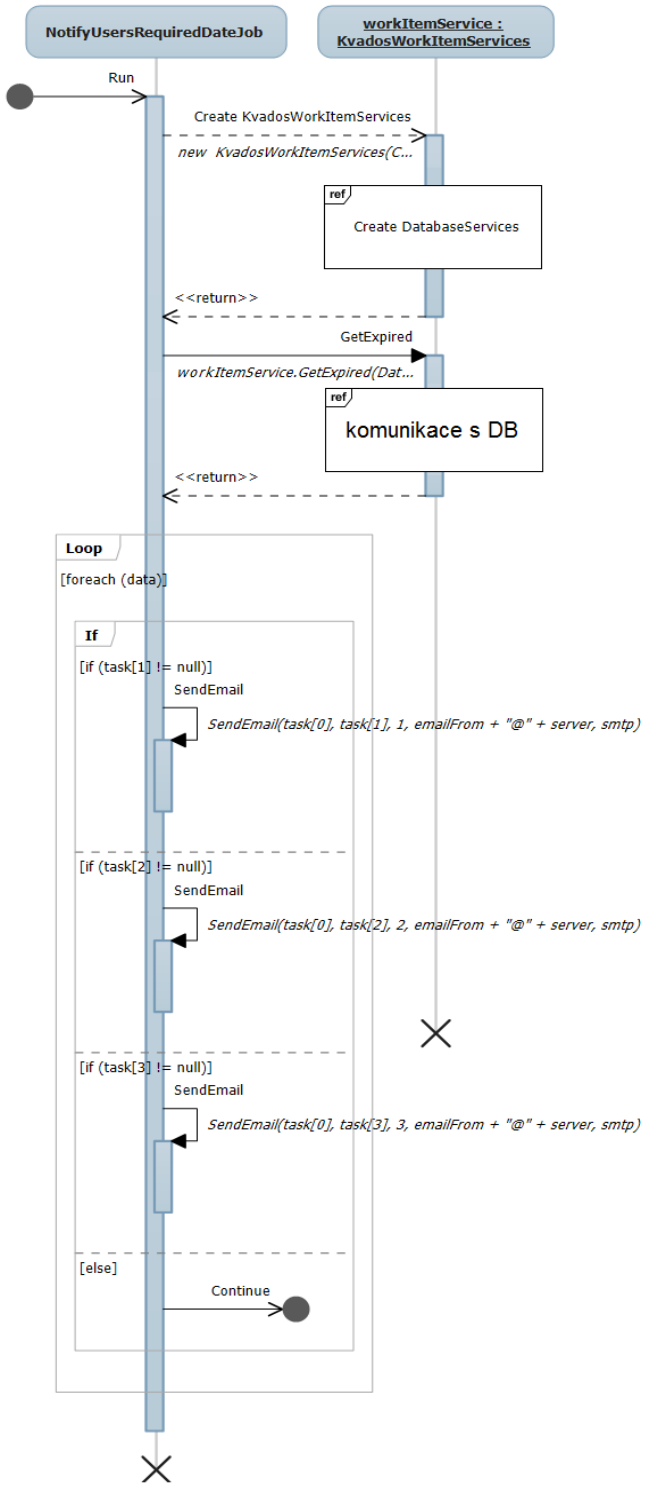
Atributy, podle nichž probíhá dotazování nejčastěji, byly rozšířeny o indexy.

5.9 Sekvenční diagramy

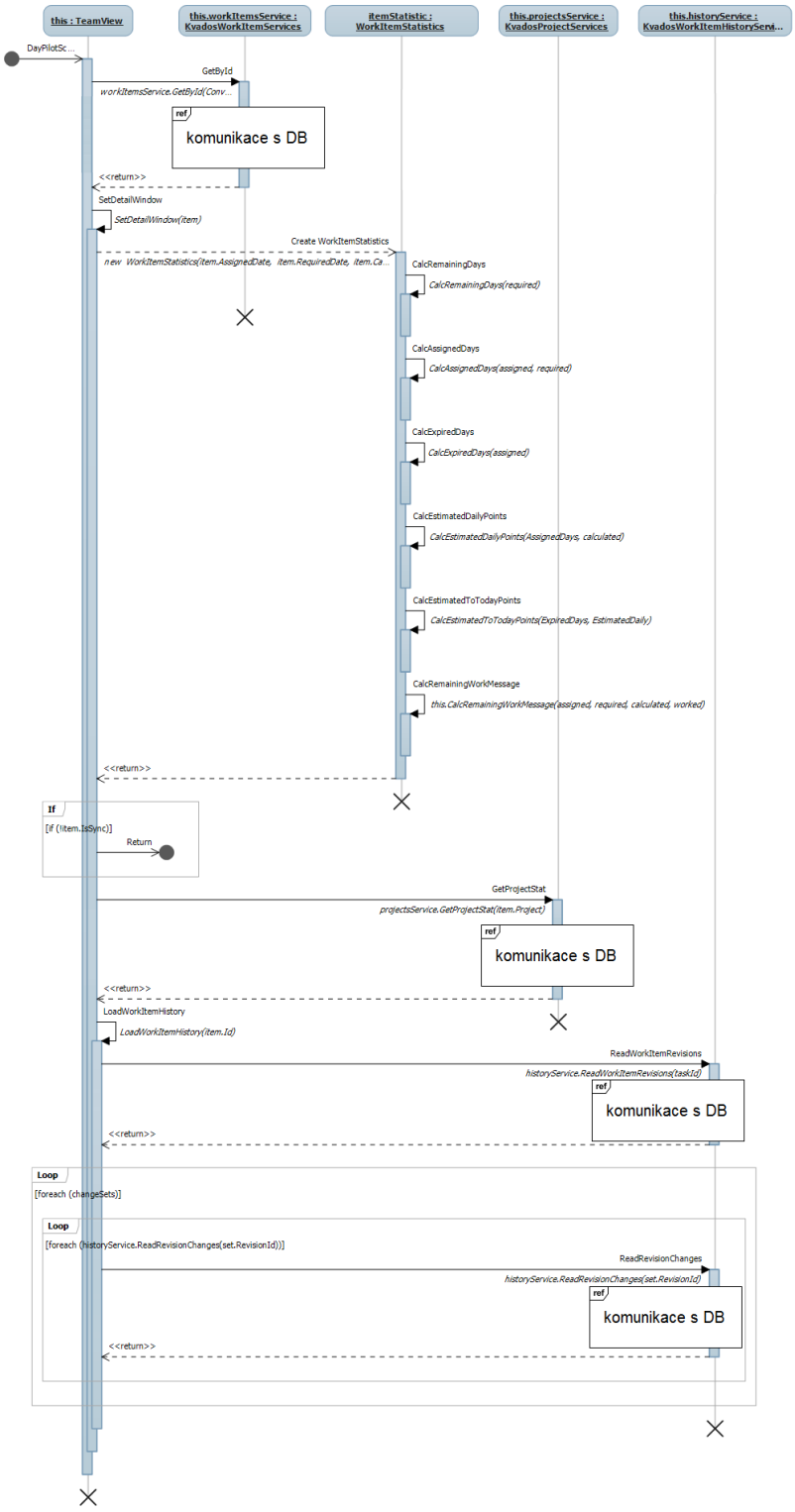
Vybrané sekvenční diagramy některých funkcí. Další lze nalézt v příloze A.



Obrázek 5.7: Sekvenční diagram funkcí 1. Rozcestník a 2. Měsíční výhled



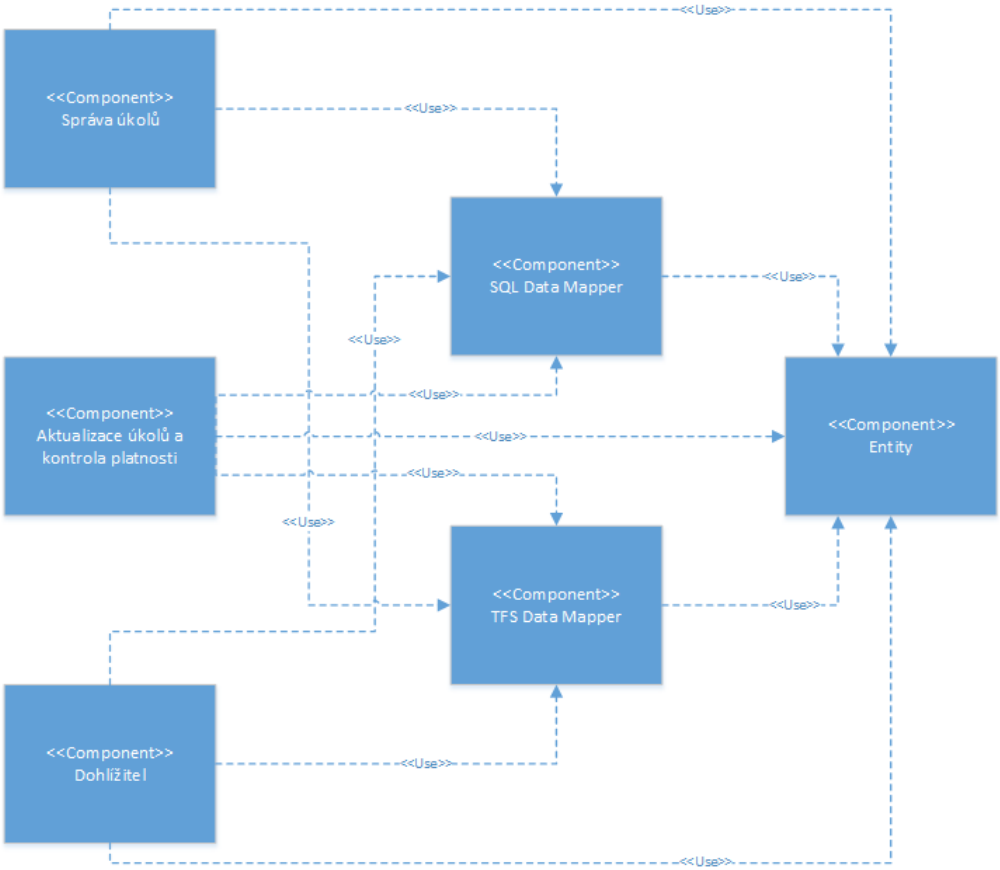
Obrázek 5.8: Sekvenční diagram funkce 13. Hlídat termín splnění



Obrázek 5.9: Sekvenční diagram 4. Detail úkolu

5.10 Diagram komponent

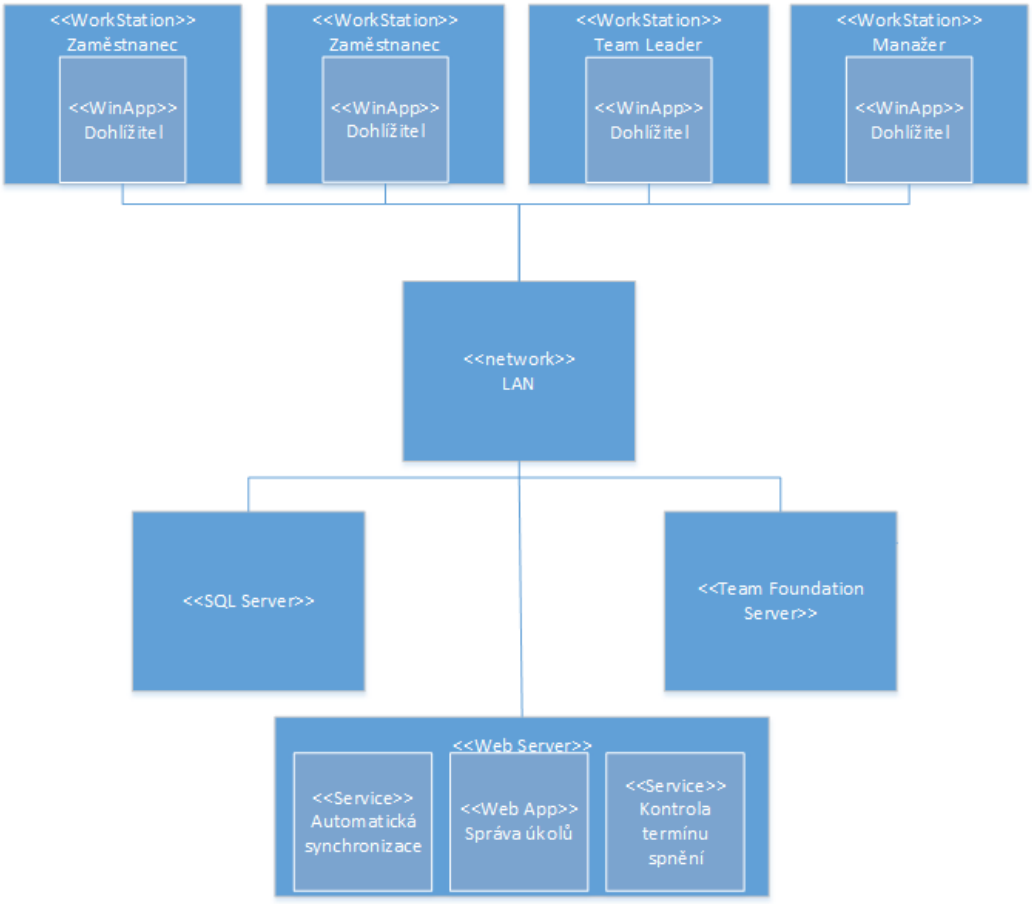
Celé řešení je logicky rozděleno do několika na sobě závislých komponent.



Obrázek 5.10: Logické rozdělení komponent

5.11 Diagram nasazení

Řešení využívá firemní lokální síť. Zaměstnanci mají každou svou pracovní stanici připojenou do této sítě. Na pracovních stanicích je nainstalována aplikace "Dohlížitel". V síti se také nachází webový server, díky němuž je přístupná webová aplikace "Správa úkolů". Pracovní stanice s ní mohou komunikovat pomocí webového prohlížeče přes zabezpečený protokol HTTPS. Jak "Dohlížitel", tak "Správa úkolů" komunikují s SQL Serverem a Team Foundation Serverem, z kterých čtou a zapisují data.



Obrázek 5.11: Schéma možného nasazení modulů v rámci firemní sítě

6. Závěr

V této práci byl vysvětlen způsob práce vývojářského týmu při tvorbě nového softwaru. Bylo představeno několik softwarových nástrojů pro usnadnění řízení týmu programátorů. Ze srovnání vyplynulo, že volba nástroje je často ovlivněna způsobem a organizací práce v týmů a že zcela univerzální řešení neexistuje.

V praktické části byly odhaleny nedostatky Team Foundation Serveru pří řízení týmu podílejícího se na více projektech. Pro potřeby zaměstnavatele bylo navrhnuto řešení pro překonání těchto odhalených nedostatků. Toho bylo docíleno formou implementace nástroje pro lepší přehled a správu úkolů týmu. Ten se skládá z hlavního webového rozhraní a podpůrných služeb, doplněného o desktopovou aplikaci pro čtení zadaných úkolů, měření odpracovaného času a jeho vykázání.

Vyvinutý nástroj pokrývá nejzákladnější okruh požadavků a je možno (jako většinu softwaru) jej v budoucnu dále plynule rozvíjet. Do finální implementace se nakonec nedostaly všechny zamýšlené funkcionality (nejsou součástí analýzy). Například byla zamýšlena funkce pro výpočet reálného produktivního času zaměstnance na základě již dokončených úkolů.

Pro autora bylo největším přínosem zpracování analýzy vyvíjeného nástroje, studium použitých technologií a jejich následné praktické využití při implementaci požadovaných funkcionalit.

7. Použitá literatura

[1] VENTUS: Software. *KVADOS software solutions* [online]. 2017. [cit. 2017-04-12]. Dostupné z: https://www.kvados.cz/produkty/ventus/

[2] Team Foundation Server 2013: Platform. *Microsoft* [online]. 2017. [cit. 2017-04-12]. Dostupné z: https://www.visualstudio.com/cs/tfs/

[3] Ext.NET: Library. [online]. 2017. [cit. 2017-04-12]. Dostupné z: http://www.ext.net

[4] Objektový model TFS: .NET assemblies. *Developer Network* [online]. 2017. [cit.2017-04-12]. Dostupné z: https://www.visualstudio.com/cs-cz/docs/setup-admin/tfs/architecture/extend-vs-for-alm

[5] SHAH, Anup. *Ext.NET Web Application Development*. Birmingham: Packt Publishing, 2012. ISBN 978-1-84969-324-0.

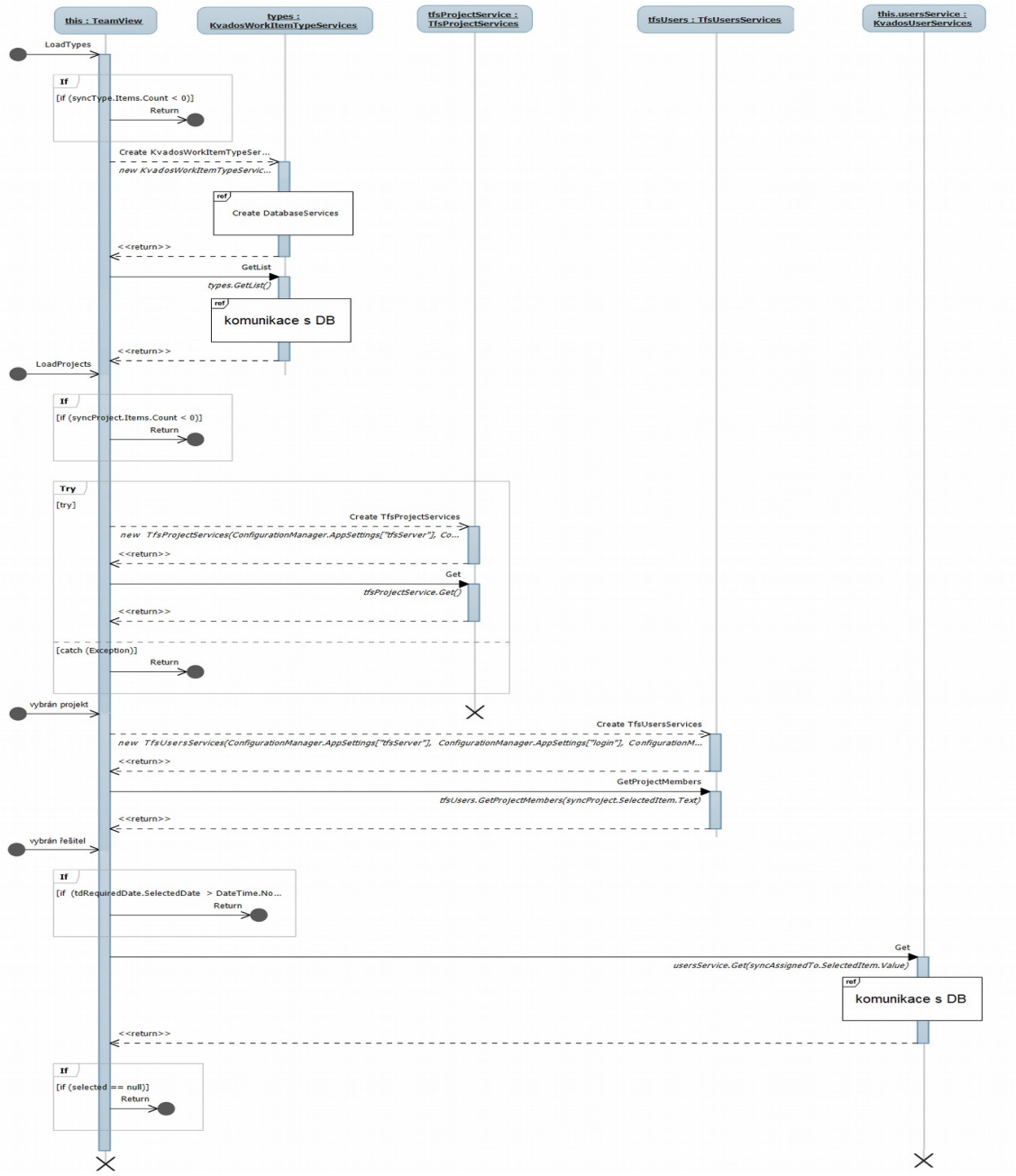
[6] Project management software, online collaboration: Basecamp. *Basecamp: Project Management & Team Communication Software* [online]. Copyright ©1999 [cit. 12.04.2017]. Dostupné z: https://basecamp.com/

[7] JIRA Software - Issue & Project Tracking for Software Teams | Atlassian. *Atlassian | Software Development and Collaboration Tools* [online]. Copyright © 2017 Atlassian [cit. 12.04.2017]. Dostupné z: https://www.atlassian.com/software/jira

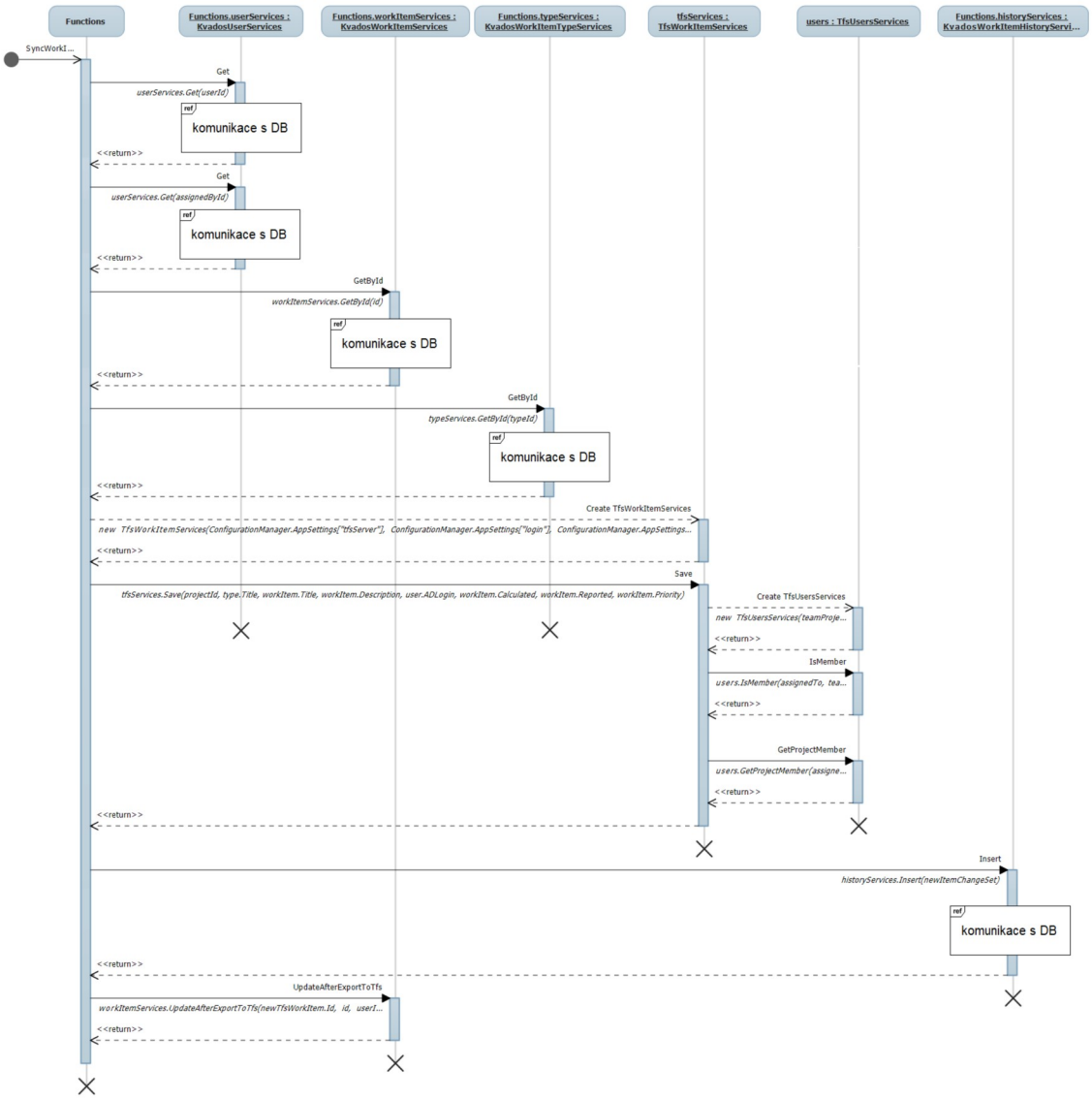
[8] Project Management and Collaboration Software | Orangescrum. *Project Management and Collaboration Software | Orangescrum* [online]. Copyright © 2011 [cit. 12.04.2017]. Dostupné z: https://www.orangescrum.com/

8. Seznam příloh

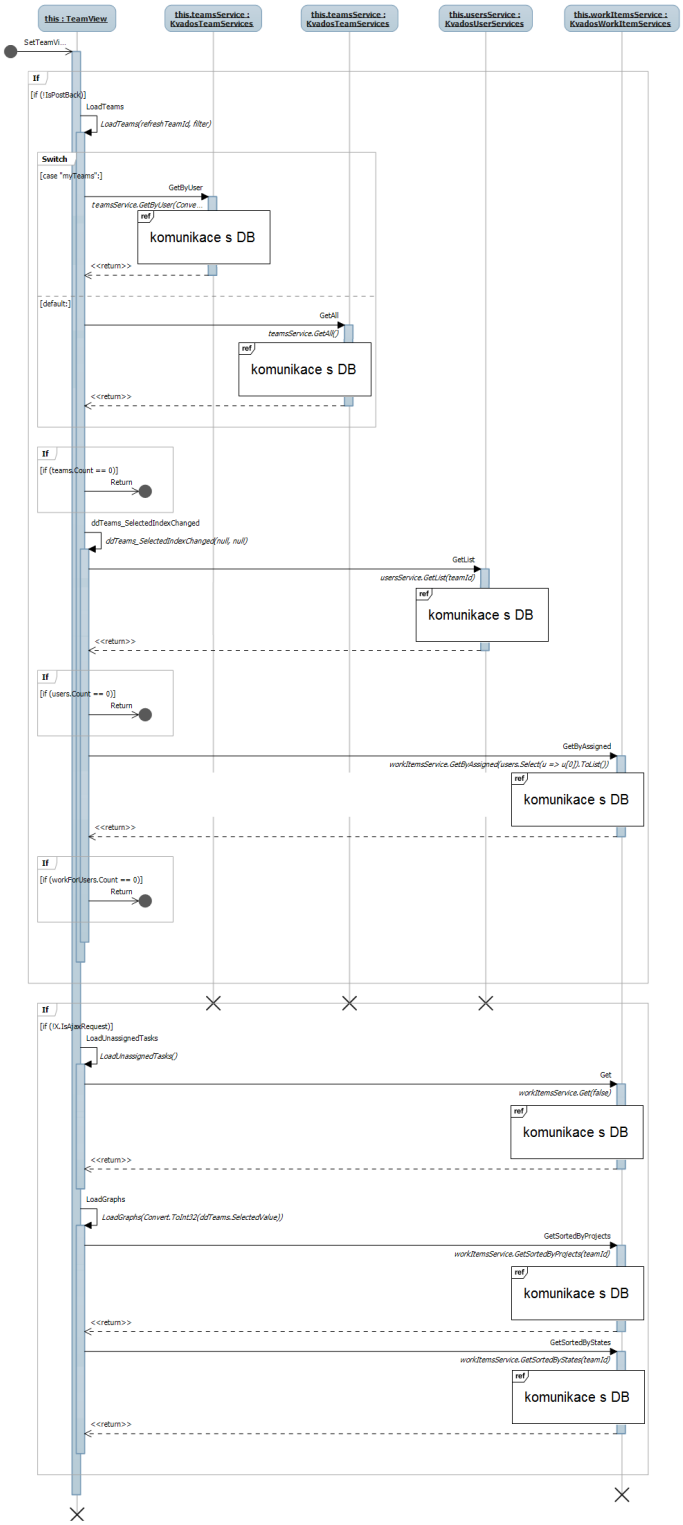
A Sekvenční diagramy



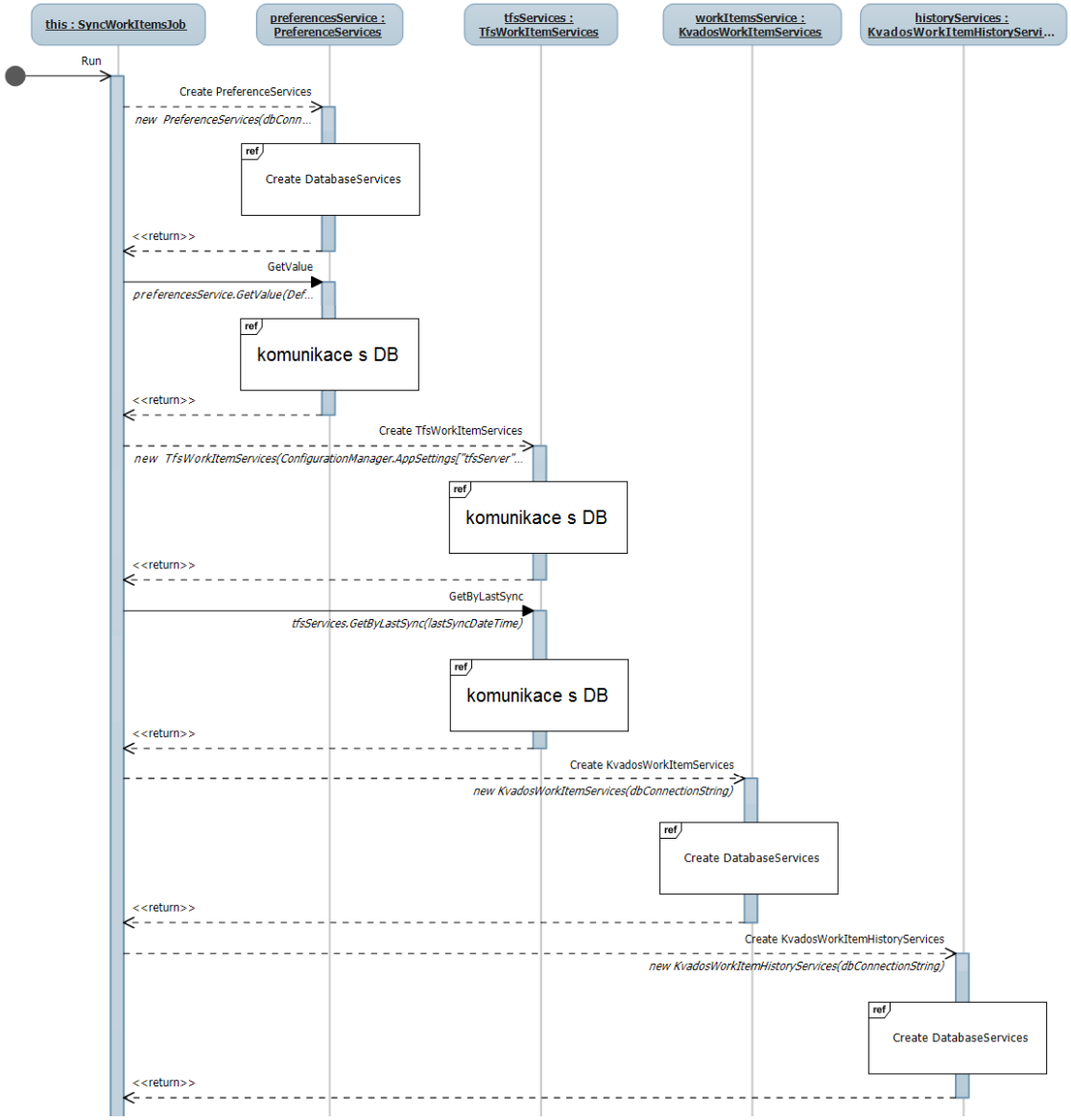
Obrázek A.1: Sekvenční diagram 8. Zadat úkol řešiteli, část I.



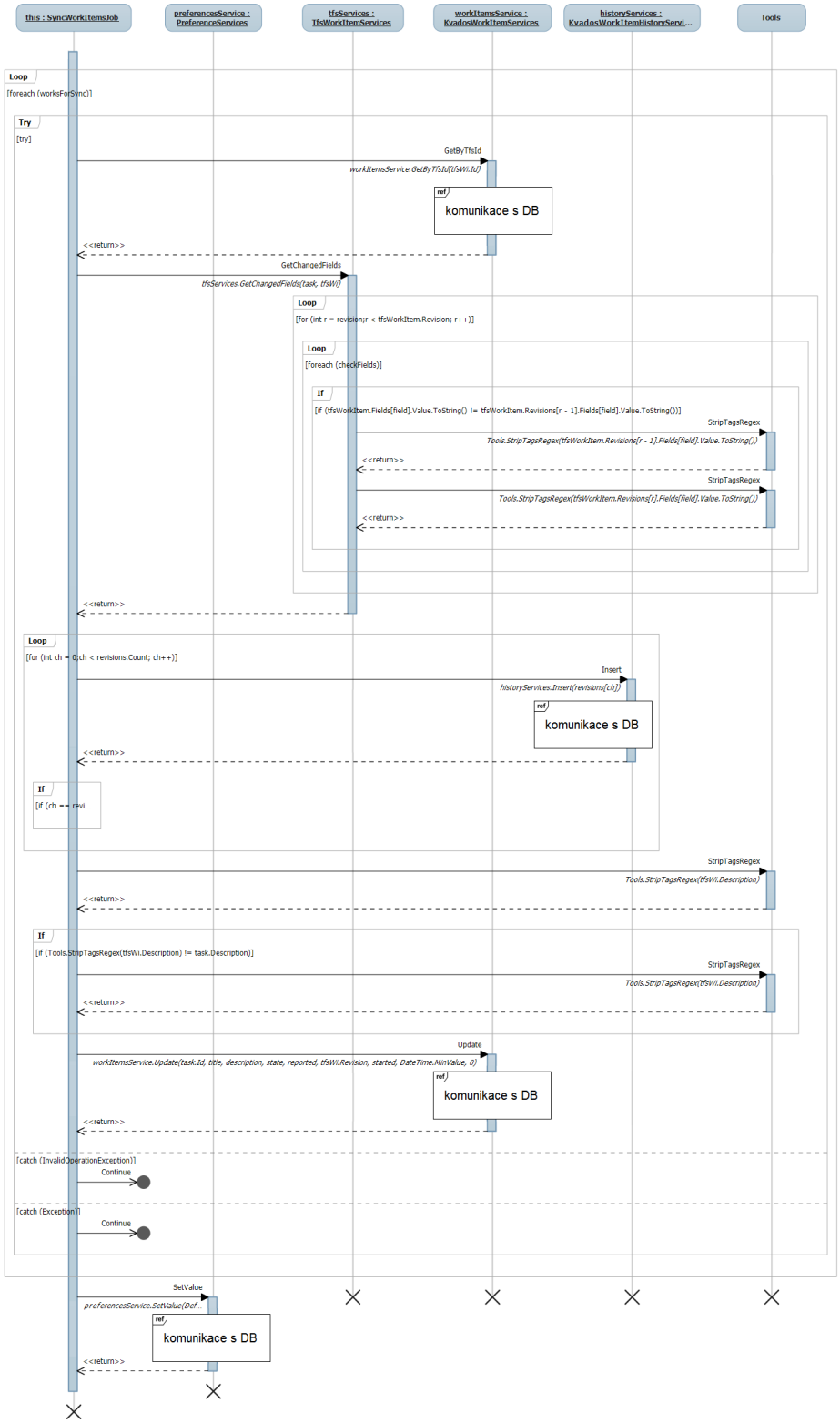
Obrázek A.2: Sekvenční diagram 8. Zadat úkol řešiteli, část II.



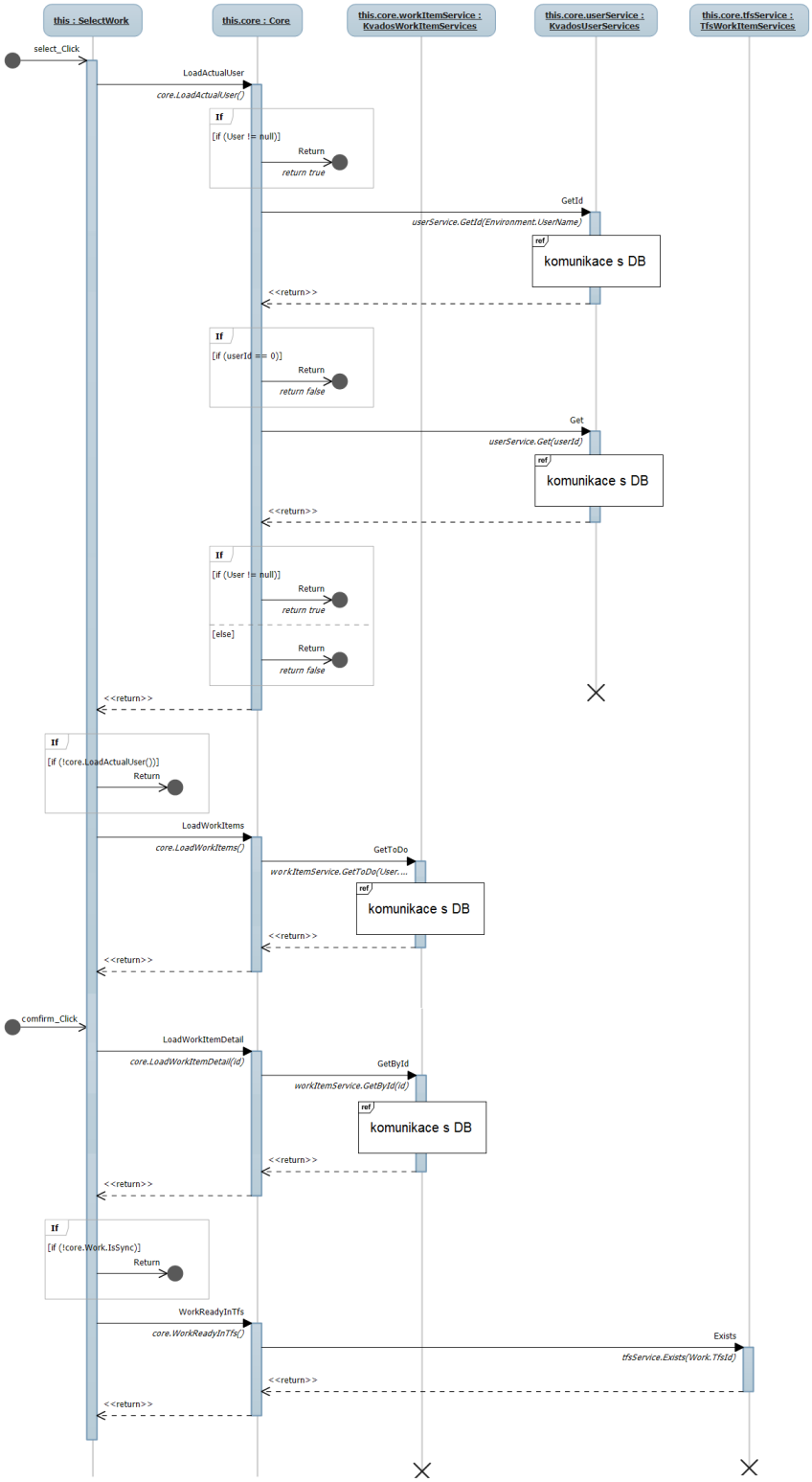
Obrázek A.3: Sekvenční diagram 17. Týmového přehledu a 18. Statistiky týmu



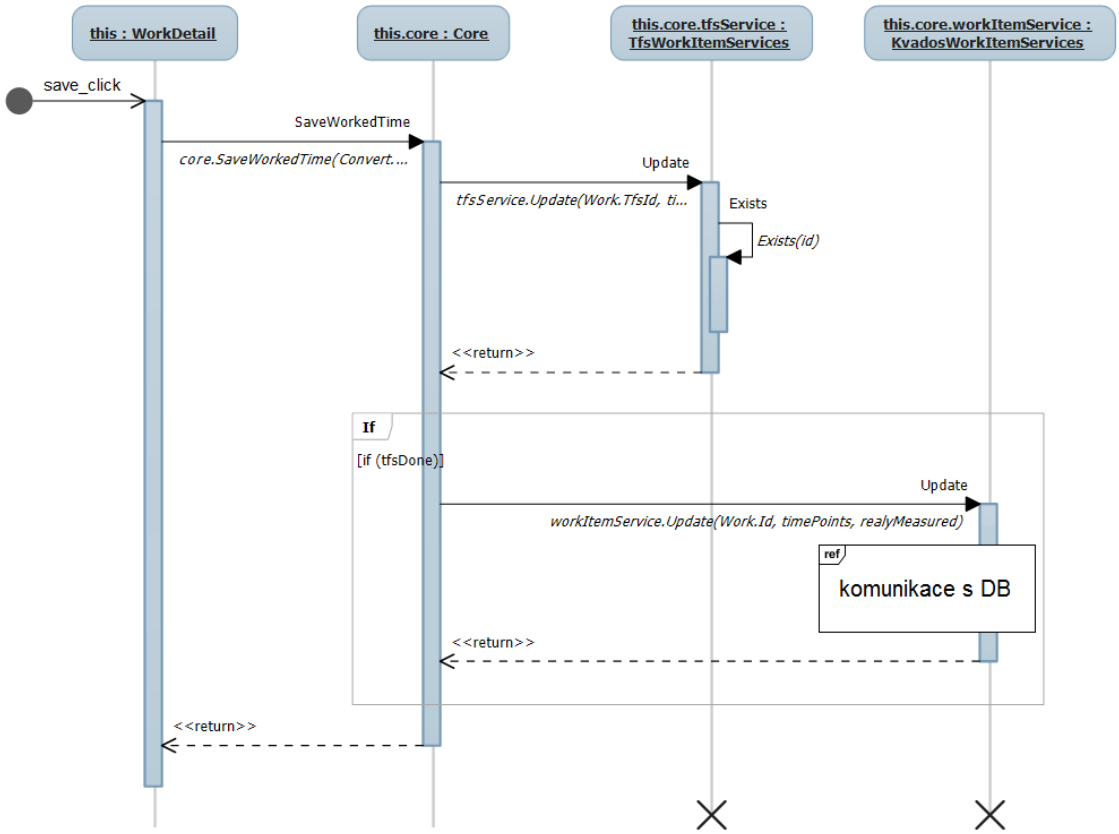
Obrázek A.4: Sekvenční diagram automatické 11. Aktualizace dat úkolu a 12. Aktualizace historie úkolu, část I.



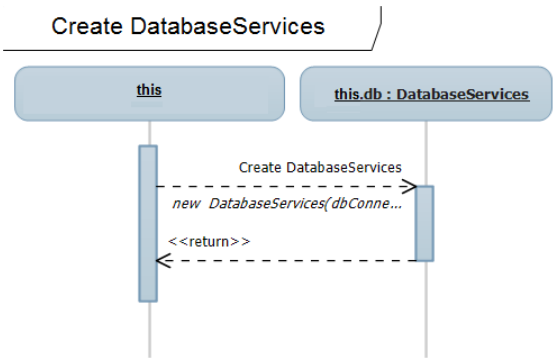
Obrázek A.5: Sekvenční diagram automatické 11. Aktualizace dat úkolu a 12. Aktualizace historie úkolu, část II.



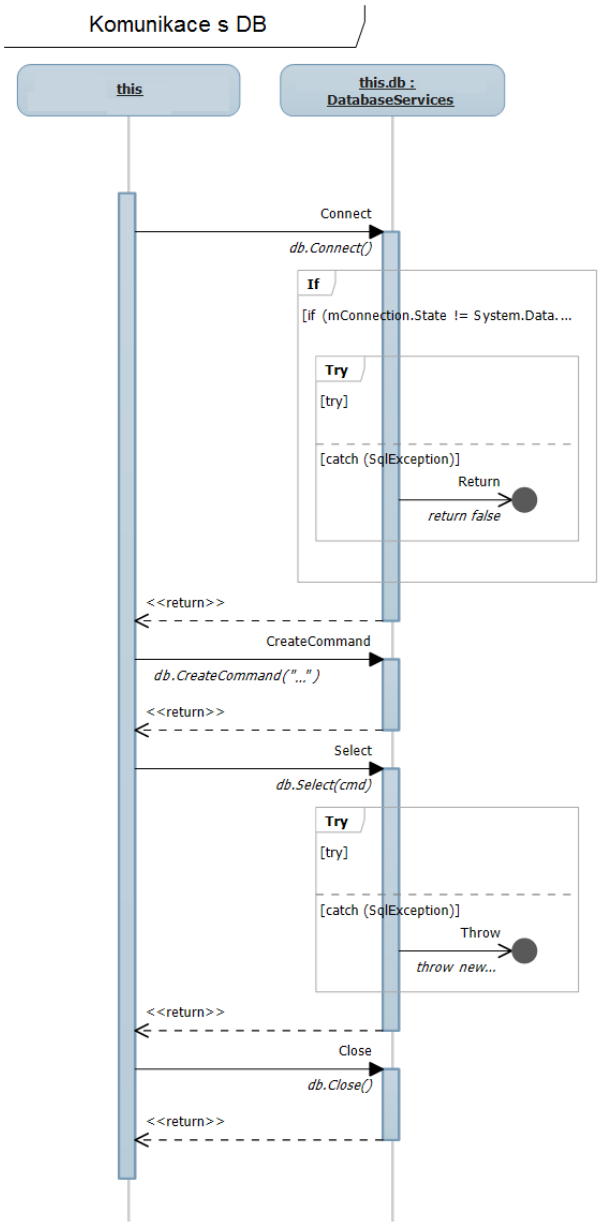
Obrázek A.6: Sekvenční diagram 20. Informace o zadaném úkolu



Obrázek A.7: Sekvenční diagram uložení 22. Vykázat odpracovaný čas

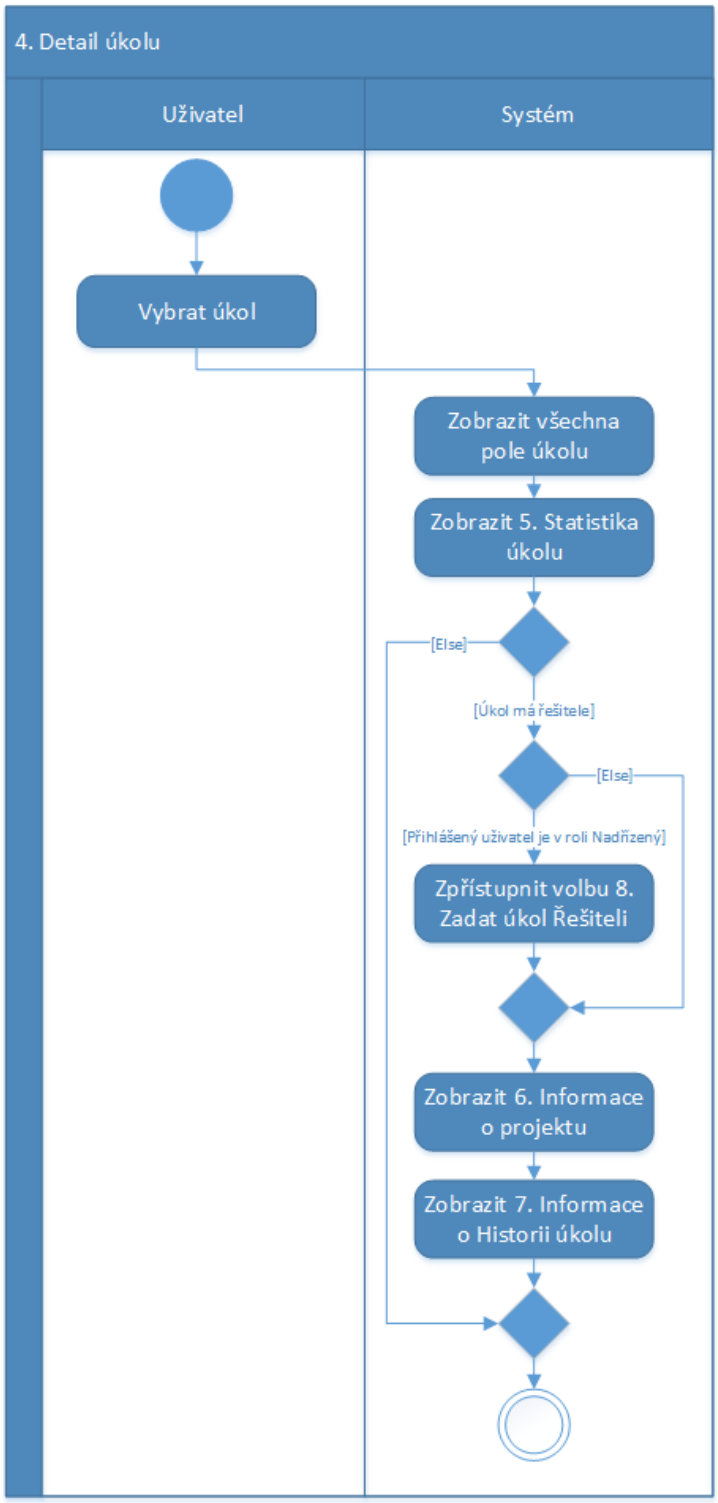


Obrázek A.8: Sekvenční diagram tvorby pomocného objektu pro práci s databází

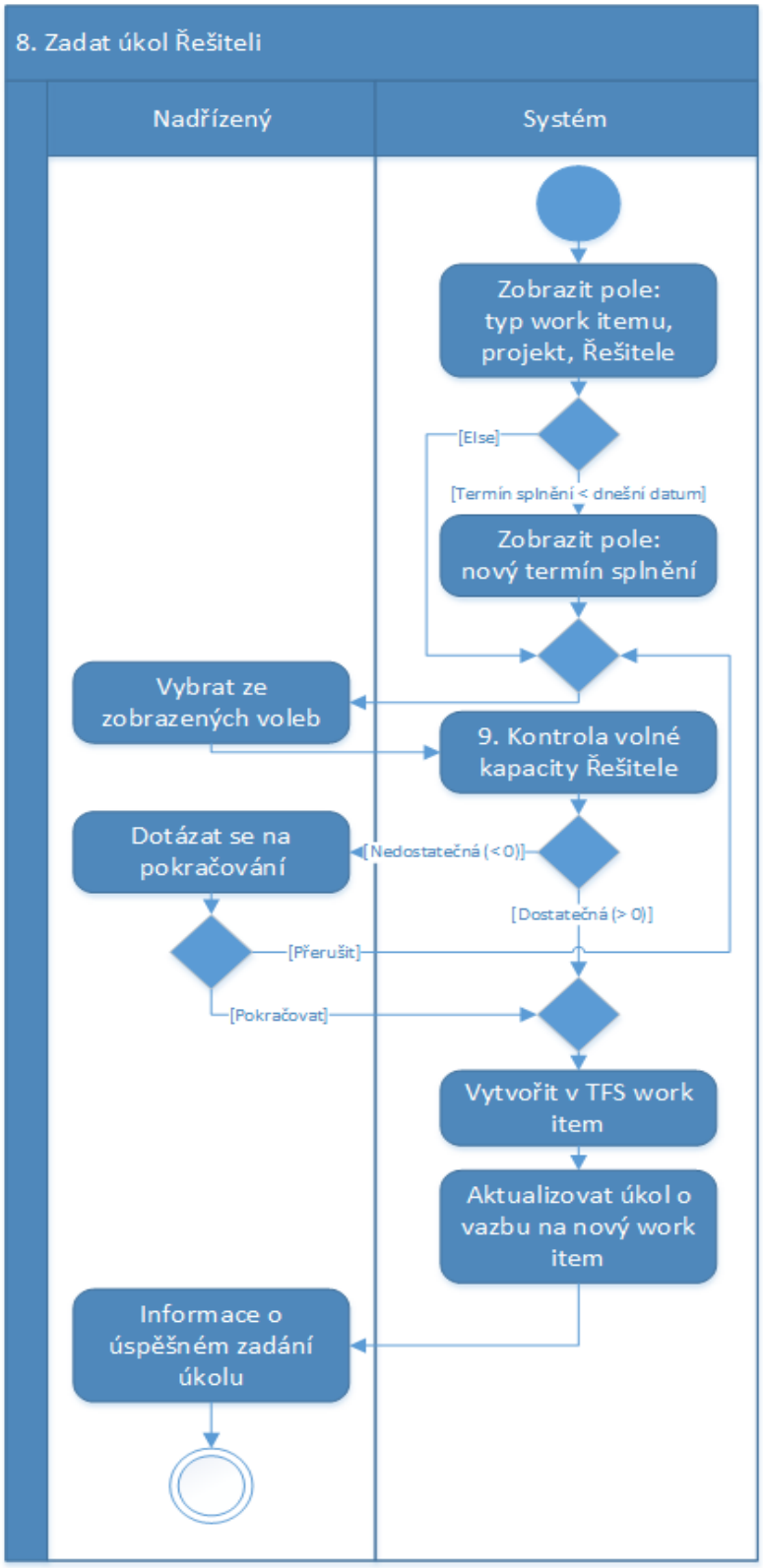


Obrázek A.9: Sekvenční diagram volání uložené procedury v databázi

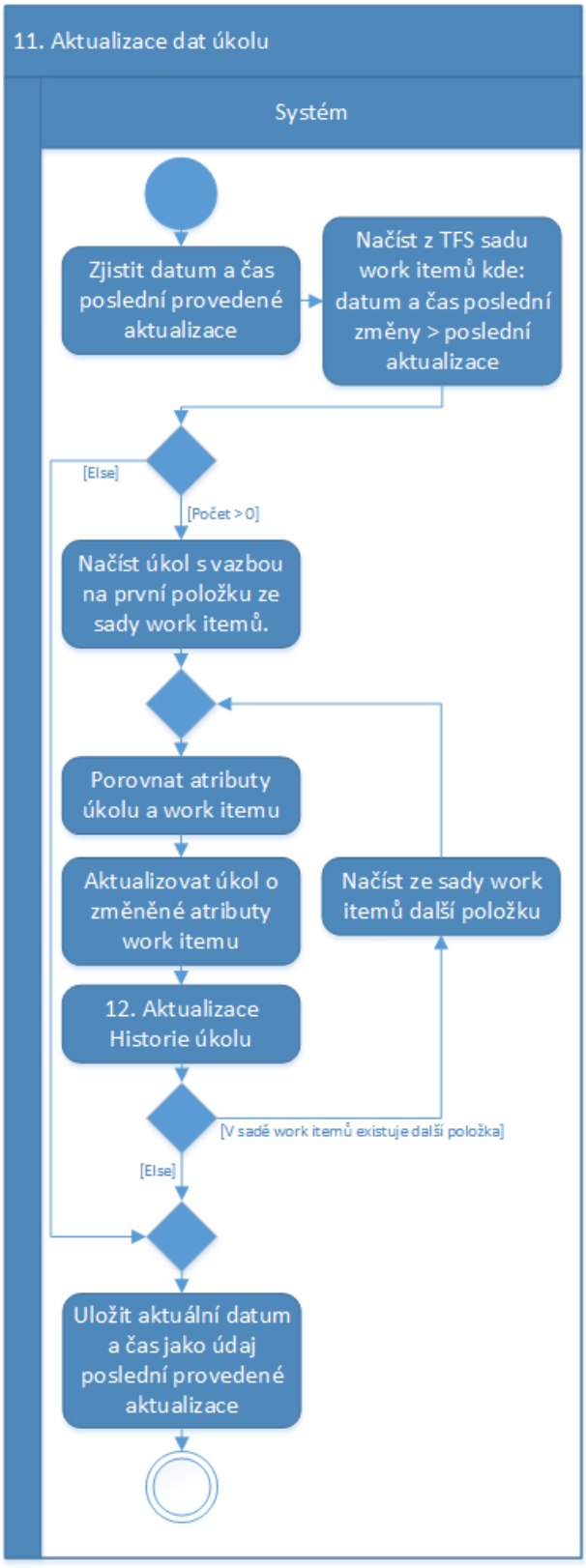
B Diagramy aktivit



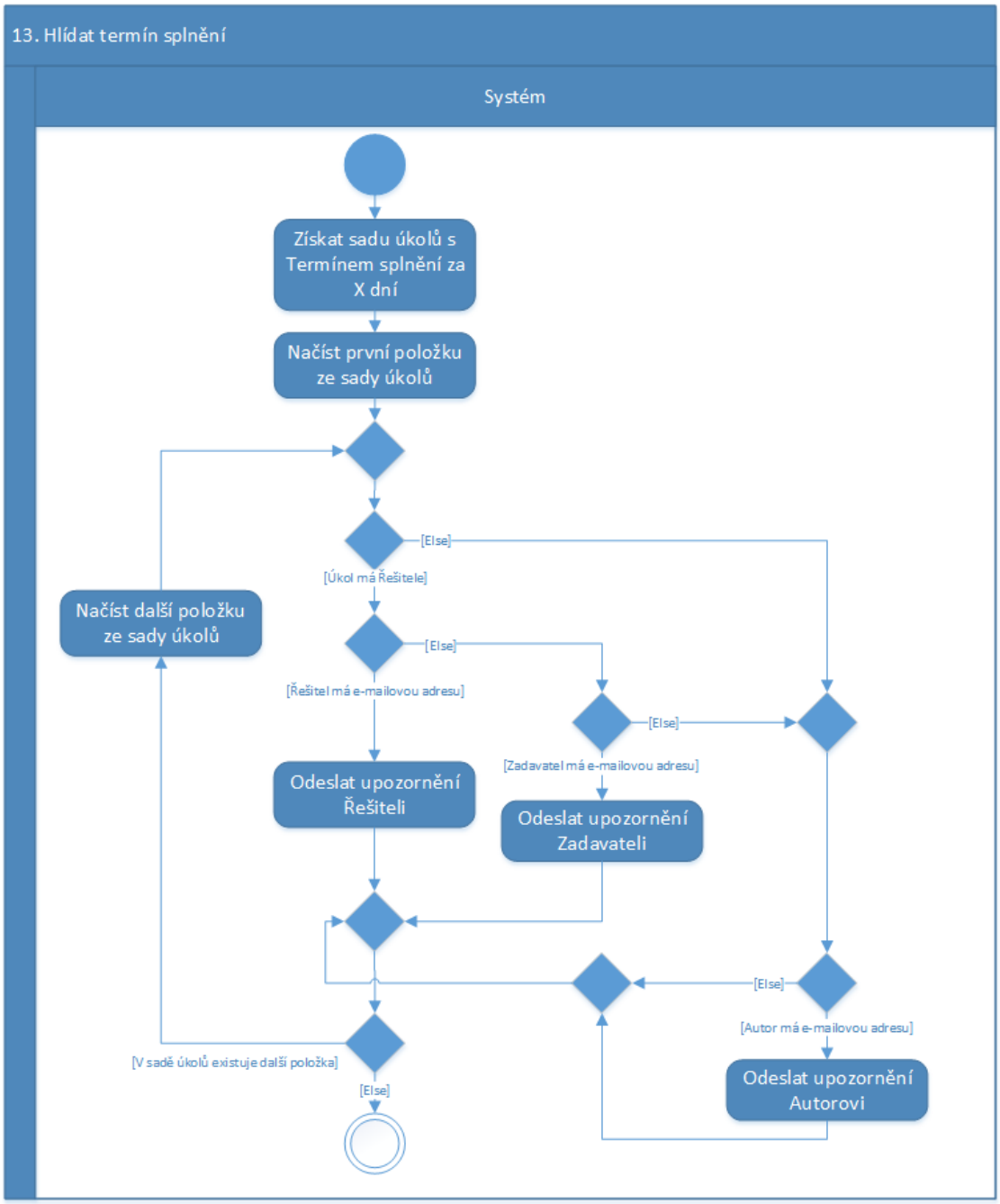
Obrázek B.1: Diagram aktivit 4.Detail úkolu



Obrázek B.2: Diagram aktivit 8. Zadat úkol řešiteli



Obrázek B.3: Diagram aktivit 11. Aktualizace dat úkolu



Obrázek B.4: Diagram aktivit 13. Hlídat termín splnění

C Vybrané detaily implementace

C.1 Práce s objektovým modelem TFS

Pro komunikaci s TFS pomocí kódu slouží jeho API. Tyto knihovny zajišťují rozhraní pro jednotný přístup k datům. Laicky jde o jakýsi ADO.NET pro TFS. Fyzicky jsou data z TFS také uložena na SQL Serveru, avšak jejich struktura je komplikovaná a nevhodný zásah by mohl způsobit nefunkčnost celého systému. Nás budou primárně zajímat třídy pro správu projektů, jeho členů a work itemů. Pro zpřístupnění těchto tříd přidáme reference na tyto jmenné prostory (Název a obsah se někdy dle verze frameworku v minulosti měnil. Testováno na .NET 4.5, reference na knihovny verze 4.0):

```
Microsoft.TeamFoundation.Client
Microsoft.TeamFoundation.Common
Microsoft.TeamFoundation.WorkItem.Tracking.Client
Microsoft.TeamFoundation.WorkItem.Tracking.Common
Microsoft.VisualStudio.Services.Client
Microsoft.VisualStudio.Services.Common
```

Jako příklad způsobu práce si ukážeme jak vytvořit jednoduchý úkol. Základ je připojit se na server a získat objekt WorkItemStore spravující work itemy:

```
Uri tfsUri = new Uri(tfsServerString);
TfsTeamProjectCollection teamProjectCollection = new
TfsTeamProjectCollection(tfsUri);
teamProjectCollection.Credentials = new NetworkCredential(user,
password, domain);
WorkItemStore workItemStore =
teamProjectCollection.GetService<WorkItemStore>();
```

Získáme objekt projektu dle jeho id. Id projektu získáme ze workItemStore.Projects.

```
Project teamProject = workItemStore.Projects.GetById(projectId);
```

Vytvoříme základ nového úkolu daného typu pro získaný projekt. Proměnná typeName obsahuje název existující šablony typu work itemu. Tvorba vlastního typu je popsána v C.4. V případě že projekt zadaný typ neexistuje, dojte k vyhození výjimky.

```
WorkItemType workItemType = teamProject.WorkItemTypes[typeName];
WorkItem newWorkItem = new WorkItem(workItemType)
{
Title = "Název úkolu",
Description = "Popis úkolu",
State = "To Do"
};
```

Zde narážíme na první problém s následným uložením, který může nastat. Pole state má pevně definované hodnoty, jenž může nabývat. V případě zadání vlastní hodnoty dojde při uložení opět k vyvolání výjimky.

Nastavení dalších polí můžeme provést následovně. Název pole zadáváme dle názvu definovaném v typu work itemu. Názvy standardních polí jsou dostupné přes výčet CoreField.

```
newWorkItem.Fields["NAZEV_POLE"].Value = "Hodnota";
```

Jestli je úkol připraven k uložení, zjistíme metodou Validate().

```
ArrayList errors = newWorkItem.Validate();
```

Je-li výsledné pole hodnot prázdné, je vše v pořádku. V opačném případě nalezneme seznam polí s popisem chyb, kvůli nimž nelze work item uložit. Doposud však není úkol v TFS uložen. To provedeme zavoláním metody Save().

```
newWorkItem.Save();
```

Pokud jsme work item správně definovali, vše se uloží do TFS a work item bude viditelný v GUI. Jestliže uložení vyvolá výjimku, musíme ve většině případů věnovat zvýšenou pozornost poli hodnot, které vrací metoda Validate().

C.2 Získání jména TFS uživatele dle doménové jména v Active Directory

Objektový model TFS pro (snadnější) nastavování uživatele obecně očekává jeho jméno v plném znění. Pokud chceme například nastavit řešitele úkolu, provedeme to následovně:

```
newWorkItem.Fields[CoreField.AssignedTo].Value = "Miroslav
Lazar";
```

To však není vhodné při komplexnějších operacích, kde je vhodnější použít klasicky unikátní identifikátor. V tomto případě ve formátu uživatelského jména z Active Directory. Následující ukázka představuje, jak získat plné jméno člena projektu dle jeho doménového jména a případné ho do tohoto projektu přidat. To z důvodu, že pokud je na work itemu nastaven uživatel, který není členem týmu, nelze work item posléze uložit.

```
public string GetProjectMember(string adLogin, string project,
bool add)

{

TeamFoundationIdentity identity;
IIdentityManagementService usersService =
teamProjectCollection.GetService<IIdentityManagementService>();
identity =
usersService.ReadIdentity(IdentitySearchFactor.AccountName,
adLogin, MembershipQuery.Direct,
ReadIdentityOptions.ExtendedProperties);
```

```
var tfsGroupIdentity =
usersService.ReadIdentity(IdentitySearchFactor.AccountName,

string.Format("[{0}]\\{0} Team", project), MembershipQuery.None,
ReadIdentityOptions.IncludeReadFromSource);

if (identity == null) return String.Empty;

if (!usersService.IsMember(tfsGroupIdentity.Descriptor,
identity.Descriptor) && add)


{                usersService.AddMemberToApplicationGroup(tfsGro
upIdentity.Descriptor, identity.Descriptor);

            }

            return identity.DisplayName;

        }
```

C.3 Dotazování se na work itemy dle data a času poslední změny

Většinu dat lze získat přes nejrůznější metody objektového modelu TFS. Pro získání kolekce work itemů splňujících specifickou podmínku podporuje TFS dotazovací jazyk s syntaxí podobnou SQL. Musíme pouze znát názvy jednotlivých atributů (polí na work itemu) a aliasy využívaných úložišť nahrazující entity z databáze. Pokud je však zapotřebí dotazovat se na work itemy dle data a času poslední změny, je zapotřebí navíc využít konstrukce pro asynchronní vykonání dotazu. Work itemy změněné po specifikovaném datu a času tedy získáváme takto:

```
string queryString = string.Format(@"SELECT [System.Id] FROM
WorkItems WHERE [System.WorkItemType] = 'Úkol Kvados' AND
[System.ChangedDate] > '{0}' AND [IsSync] = 1 ORDER BY
[System.Id]", utcLastSync);
Query queryWithState = new Query(workItemStore, queryString,
null, false);
ICancelableAsyncResult car = queryWithState.BeginQuery();
WorkItemCollection works = queryWithState.EndQuery(car);
```

Proměnná utcLastSync je časový údaj ve tvaru například: „2017-02-05T12:05:56“. Čas je v UTC a TFS si sám dopočítá lokální časový posun.

C.4 Úpravy šablony projektu a vlastní typy work itemů

Aby bylo možné v TFS projektech používat vlastní typy work itemů, je zapotřebí editovat šablonu projektu a rozšířit jí o nový typ work itemu. Pro úpravu šablon obsahuje Visual Studio několik nástrojů.

Pokud chceme projekt rozšířit o nový typ work itemu, postupujeme následovně:

1. Ve Visual Studiu se přes „TEAM → Connect to Team Foundation Server“ standardně připojíme k TFS, kde chceme vlastní work itemy vytvářet.
2. Ta samá nabídka „TEAM“ se rozšíří o „Team Project Collection Settings → Process Template Manager“. Zde zvolenou šablonu stáhneme.
3. Nyní budeme provádět změny lokálně. Přes „TOOLS → Process Editor → Process Templates → Open Process Template“ otevřeme staženou šablonu.
4. Zobrazí se nám uživatelské rozhraní pro editaci šablony projektu. Můžeme si tak nastavit výchozí stav našich budoucích projektů po jejich založení. Nás v tuto chvíli ale bude zajímat položka „Work Item Tracking → Type Definitions“. Zde je seznam všech typů work itemu, které lze v projektu vytvořit. Klikneme na „New“ a pojmenujeme si náš nový typ work itemu. „Copy From“ zvolíme stávající typ work itemu, z jehož kopie budeme vycházet. Nejčastěji to bude asi „Task“.
5. Po potvrzení nám v seznamu přibude náš nový typ a dvojklikem ho otevřeme. Vidíme seznam všech polí tohoto typu work itemu. My potřebujeme evidovat vlastní data, klikneme proto opět na „New“ pro přidání vlastního pole a zadáme vše potřebné. Pro odlišení vlastních a systémových polí můžeme do „Reference Name“ zadávat vlastní prefix dle vlastní šablony. Například „NAZEVFIRMY.POLE“.
6. Přidání poli automaticky neznamená, že půjdou na detailu work itemu také vidět. Doplnění sloupce do uživatelského rozhraní provedeme pravým kliknutím na nově vytvořený nebo vybraný Column (sloupec) a zvolíme „New Control“. Control si upravíme podle potřeby a před připravenou definici vybereme pomocí „Field Name“. Tady přijde vhod dřívější odlišení pole pomocí názvu jeho reference. Vzhled detailu work itemu můžeme zkontrolovat přes tlačítko „Preview Form“.
7. Změny uložíme klasicky ikonou diskety.
8. Šablonu máme připravenou. Tu již stačí pouze nahrát na TFS. Opět přes Visual Studio a nabídku „TEAM → Team Project Collection Settings → Process Template Manager“. Zde originální šablonu smažeme, případně pro jistotu zase stáhneme, kdyby naše upravená šablona nešla z nejrůznějších důvodů nahrát.
9. Až tak učiníme, tlačítkem Upload vyhledáme naší dříve upravenou šablonu a vyčkáme na její nahrání.
10. Po úspěšném nahrání vytvoříme přes Visual Studio nový týmový projekt a jako šablonu vybereme tu s našim vlastním typem work itemu. Při vytváření nového work itemu by nám TFS nyní mělo nabídnout náš nový typ work itemu.